Schizo Programmer's Reference Manual

Release Date: September 30, 2007

Products Rights Notice:

Schizo Overview 1

1.1 Introduction

The Schizo chip is the primary connection between the Safari Bus and the I/O Subsystem. Schizo features include:

- Full master and slave port connection to the high-speed Safari bus. Safari is a split address / data packet-switched bus:
  - Schizo provides controls for external switch required to reduce 256-bit Safari data bus to 64-bit Schizo connection.
  - Maximum data throughput of 1.2 Gbyte/sec.(150MHz Safari)
  - Safari data is ECC protected.
- A UPA64S leaf, controlling a high-speed, slave only, UPA bus segment:
  - UPA Datapath external to Schizo (Schizo provides switch controls)
  - Compliant with UPA Specification Revision 1.1
  - Support for two slave devices
  - Sustainable write data throughput of 800Mb/sec.
- Two independent PCI leafs, each controlling a single PCI bus segments, each with full master and slave support:

  PCI Bus A has the following features:
  - 5 volt or 3.3 volt signalling
  - 64-bit data bus (also supports 32-bit devices)
  - Compatible with the PCI Specification (Revision 2.1)
  - PCI arbiter with support for up to six master devices

  PCI Bus B has the following features:
  - 5 volt or 3.3 volt signalling
  - 64-bit data bus (also supports 32-bit devices)
  - Compatible with the PCI Specification (Revision 2.1)
  - Compatible with the PCI 66MHz extensions (3.3 volt only)
  - PCI arbiter with support for up to two master devices

  Additionally, each PCI leaf provides the following services:
  - A 16-entry streaming cache for accelerating some kinds of PCI DVMA activity.
  - An IOMMU with 16-entry TLB for mapping DVMA addresses.

- A “Mondo-Vector” Dispatch Unit, or MDU for delivering interrupt requests to a CPU module. Each MDU provides support for several internal interrupt sources, external PCI interrupts from up to six total slots (or external interrupts from standard on-board IO devices in place of one of the slots). External interrupts are delivered from a separate external interrupt concentrator via a simple 7-bit interface.

1.2 Schizo Summary

1.2.1 Technology

- 0.35 micron, 3 level metal, 3.3 volt optimized CMOS standard cell library.

1.2.2 Package

- The Schizo die has 639 signal pads and 364 VSS/VDD pads for an approximate total pad count of 1002.
- Schizo uses a flip-chip technology, and be packaged in a 1012 ball PBGA

1.2.3 Design Size

- 820+K gates
- 164K bits RAM

1.2.4 Custom Cells

The following non-standard cells are used in the Schizo design:

- Universal (5V/3.3V compliant) PCI I/O buffers
- 66MHz capable PCI I/O buffers
- Safari DTL I/O buffers
- PLL and PECL receiver for Safari clock
- PLL and PECL receiver for UPA clock
- PLL for each PCI leaf clock

1.2.5 Frequency of Operation

- Safari operation up to 150 MHz (6.7 ns) (may be reduced).
- UPA operation up to 120 MHz (8.33 ns).
- PCI bus A operation up to 33 MHz (30 ns) (internal logic at 66 MHz)
- PCI bus B operation up to 66 MHz (15 ns)

1.2.6 Power Consumption

- Maximum power consumption: 15 Watts

1.3 *Design and Performance goals*

Schizo is designed for high-speed data transfers between the Safari bus and each leaf block (UPA, PCI-A, PCI-B). Table 1-1 shows the performance goals for each leaf block considered independently. When multiple leaf blocks are active simultaneously, performance can be less. Also .

Table 1-1 Schizo Performance **Goals**

Leaf	Operation	Latency	Throughput
UPA	PIO Read single (16 bytes)		
UPA	PIO Write single (16 bytes)	350 ns	800 Mb/sec
UPA	PIO Read block (64 bytes)		
UPA	PIO Write block		800 Mb/sec
PCI	PIO Read single		40 Mb/sec
PCI	PIO Write single		100 Mb/sec
PCI	PIO Read block		130 Mb/sec
PCI	PIO Write block		285 Mb/sec
PCI	Consistent DVMA Read	using 170 ns memory latency	76 Mb/sec
PCI	Consistent DVMA Write	using 170 ns memory latency	358 Mb/sec
PCI	Streaming DVMA Read	using 170 ns memory latency	204 Mb/sec
PCI	Streaming DVMA Write	using 170 ns memory latency	392 Mb/sec

These numbers are maximum performance estimates, and are based on the following assumptions. Performance will degrade in cases where some of the assumptions are not met.

- Only one leaf block is active at a time
- Associated leaf block is operating in highest performance mode (e.g. 66.7MHz, 64-bit, no wait states for PCI)
- All necessary lookups hit (IOMMU, STC)
- Specific memory system performance: For the above estimates, 170 ns memory latency was used (Typical of an excalibur system). For larger systems with bigger memory latencys dma performance will be less then the numbers listed above.
- No additional load on Safari bus

1.4 Schizo External Interfaces

Table 1-2 summarizes the signal pins required for the external interfaces of Schizo.

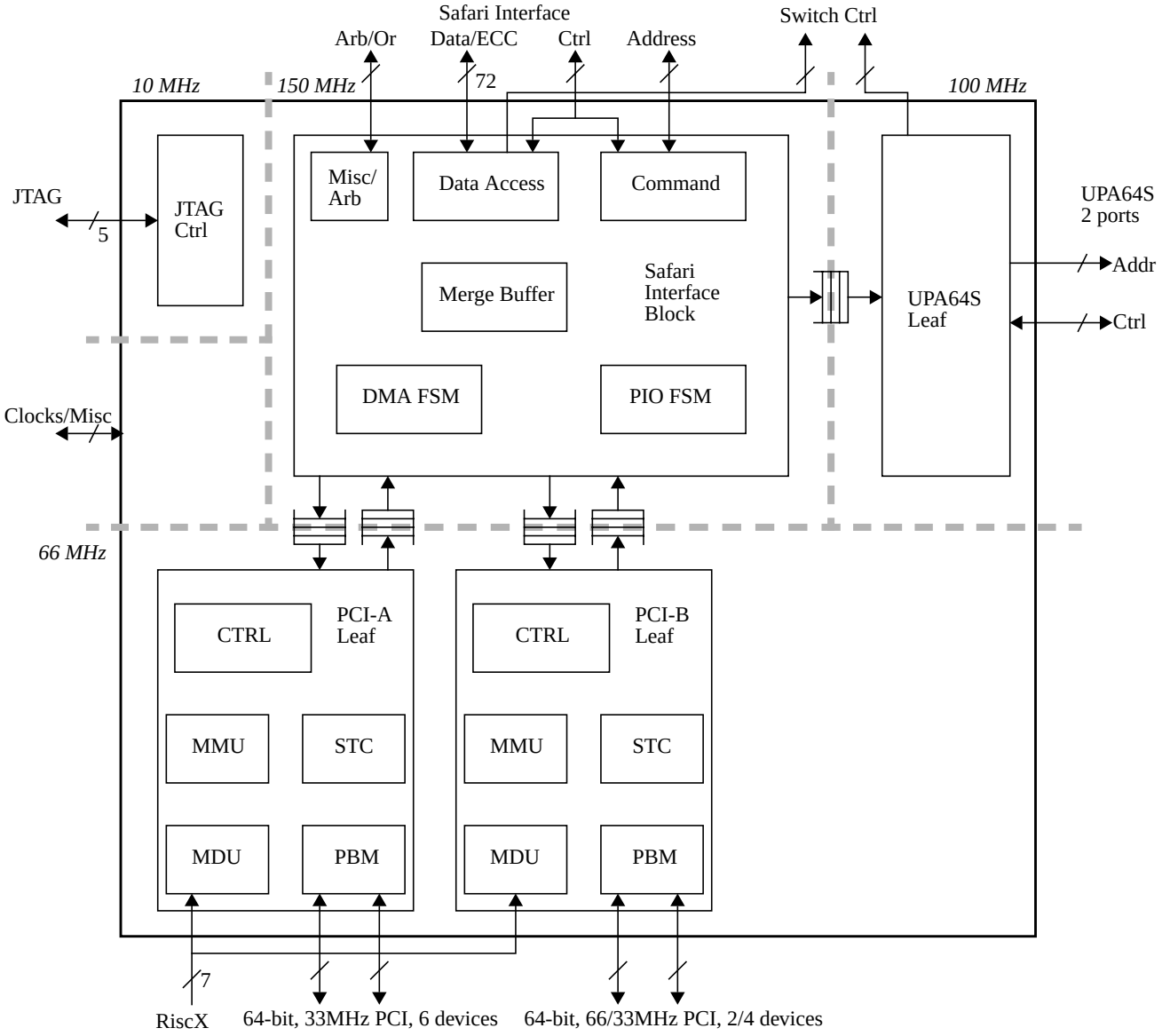
Table 1-2 Schizo External interface summary

Interface	Pins	Notes
Safari bus Interface	124	Safari 64 bit interface
Schizo specific	130	Schizo specific data switch bus
Schizo Safari debug	0	Schizo Safari debug pins are multiplexed on the Z_AID lines and Z_SYSCODE lines
UPA Interface	107	UPA Slave 64 bit interface
PCIA Interface	105	66MHz, 64 bit PCI interface
PCIA debug	6	66MHz, 64 bit PCI debug signals
PCIB Interface	124	33MHz, 64 bit PCI interface
PCIB debug	6	66MHz, 64 bit PCI debug signals
Safari PLL	9	bypass,clkout,tst,vdd,vss,clkneg,clkpos, bias,bias_vss
UPA PLL	9	bypass,clkout,tst,vdd,vss,clkneg,clkpos, bias,bias_vss
PCIA PLL	7	bypass,clkout,vdd,vss,leaf_clk, bias,bias_vss
PCIB PLL	7	bypass,clkout,vdd,vss,leaf_clk, bias,bias_vss
Test	5	JTAG port and diagnostics
Signal Total	639	
Safari	232	64-bit datapath

1.5 Schizo Block Diagram

Figure 1-1 shows a conceptual block diagram of Schizo.

Figure 1-1 Schizo Conceptual Block Diagram



1.6 Schizo Block Overviews

1.6.1 Safari Interface Overview

1.6.1.1 Introduction

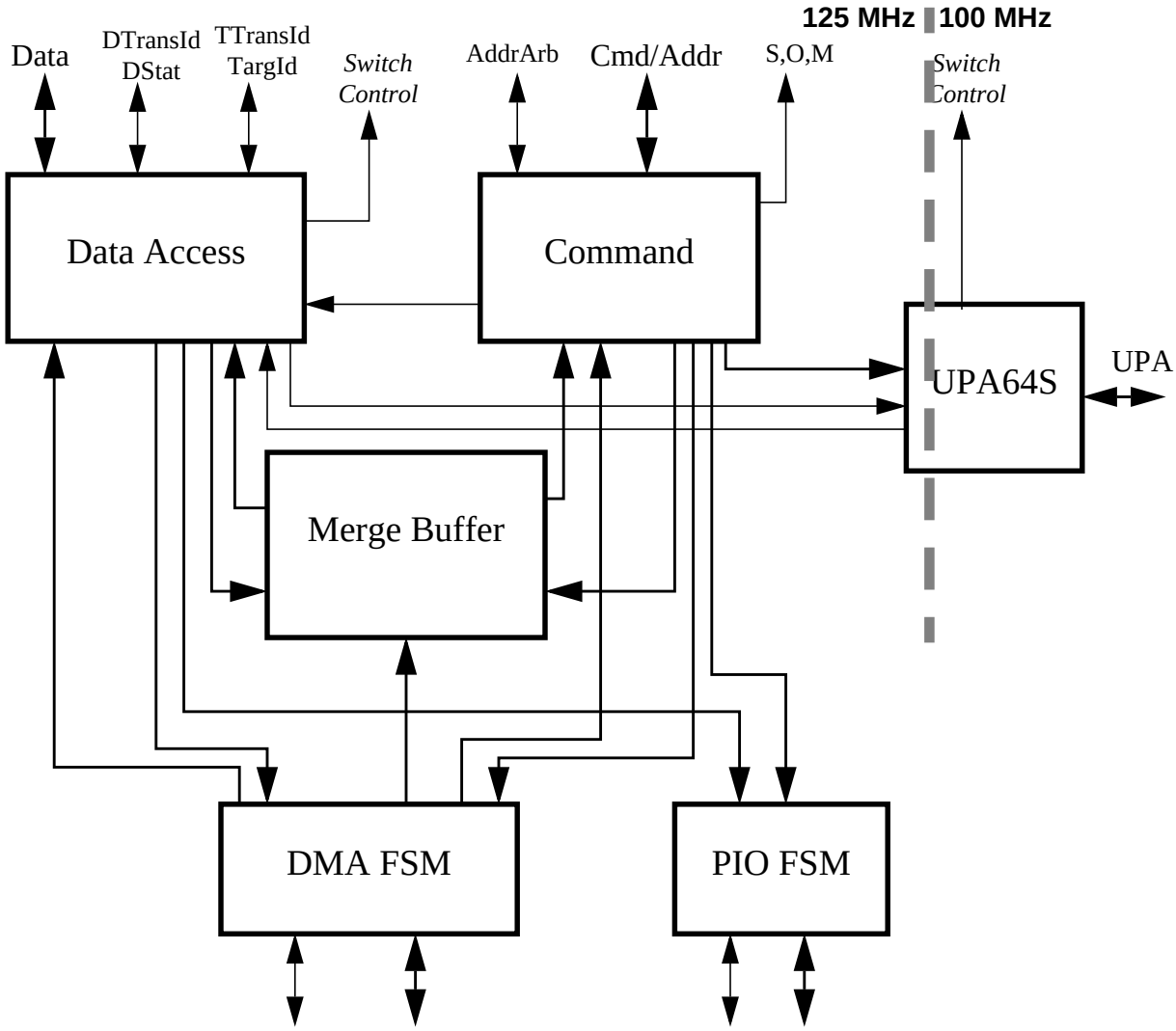
The Safari Interface provides the Safari specific functionality for the Schizo I/O chip. This major block provides a consistent internal interface for multiple “leaves”, implements the caching necessary to merge partial writes into coherent memory, retries transactions when necessary for completion, and orders out-of-order data when required.

Provision is made for three standard leaf ports, nominally two PCI and a NewLink controller. It also provides an optimized slave-only interface for UPA64S graphics accelerators.

1.6.1.2 Block Diagram

The blocks in the top level are organized functionally. The Command block deals with the Safari Transaction Request group, i.e. commands and addresses. It includes the bus or DTags (“dual” tags) for snooping, the command decoding logic and the mapping, or address range, comparators. The Data Access block, deals with the Safari Data and Data Transfer groups as well as the control of the Safari and Schizo ports of SDS (Schizo Data Switch.) The Merge block implements the CTags, or local cache tags, and the data store used for coherent writes. The DMA FSM and PIO FSM’s keep track of unfinished transactions. They also manage the command and data flow to and from the leaves.

Figure 1-1 Safari Interface Top Level Block Diagram



1.6.1.3 Sub-Block Descriptions

1.6.1.3.1 Command Block

The Command block contains the Safari snoop and address decode logic. It must sustain a one command per cycle rate and meet fixed latencies on its outputs. Once identified as accessing this Schizo, or its UPA, commands are queued to await more leisurely processing.

1.6.1.3.2 *Data Access Block*

The Data Access block takes care of transferring data to or from Schizo on Safari. It controls the Safari and Schizo ports of SDS (Schizo Data Switch) and implements the TTransId / TargId logic.

1.6.1.3.3 *Merge Buffer*

The Merge Buffer implements the coherency necessary to execute partial block writes in a Safari based system. It will be use for DMA write transactions of less than 64-bytes addressing coherent memory. While technically a cache, the merge buffers are used only for writes, not to improve the performance of read operations.

1.6.1.3.4 *Direct Memory Access Finite State Machine Block*

The DMA FSM block tracks and reassembles outstanding DMA operations for the leaf cells. It also manages the SI side of up-bound fifo in the asynchronous Internal Interface.

1.6.1.3.5 *PIO Finitie State Machine Block*

The PIO FSM block tracks PIO operations sent to the leaf cells. It translates Safari packets to Internal packets, it manages the SI side of the down-bound asynchronous fifo in the Internal Interface, and it implements transaction level flow control for the leaves.

1.6.2 *PCI Leaf Overview*

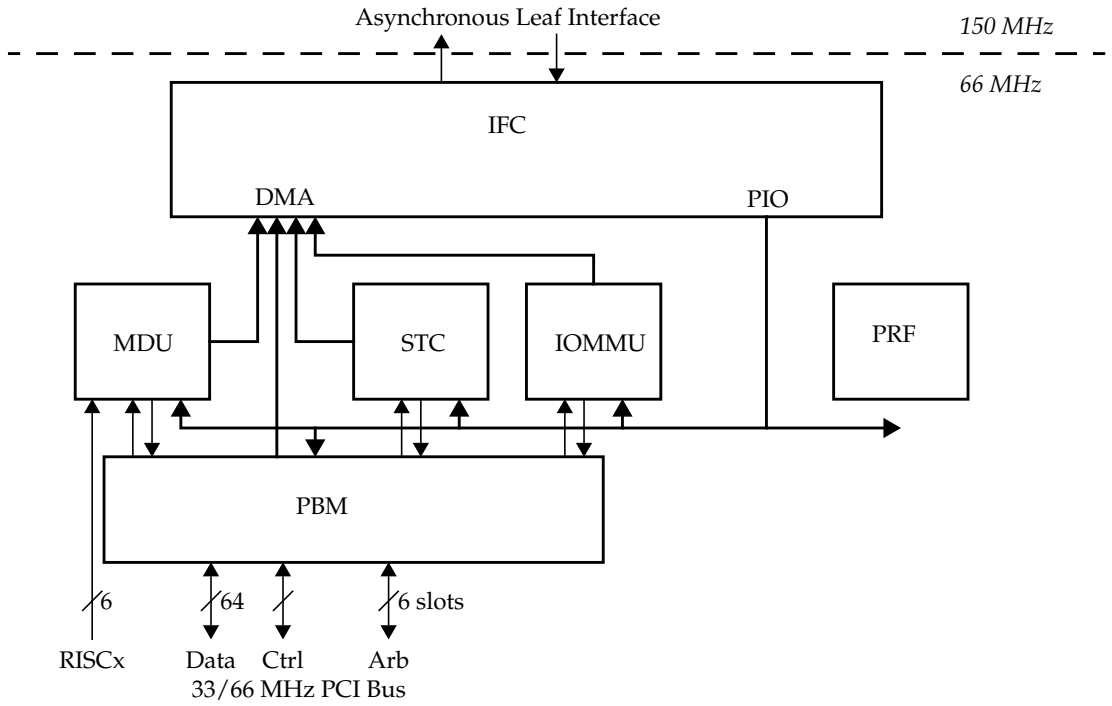
1.6.2.1 *Introduction*

The PCI leaf communicates between the internal interface to the Safari block, and a single PCI bus segment. The main purpose of the PCI leaf is to allow PIO and DMA operations to/from the PCI bus, as well as interrupts from PCI devices. In support of the PCI bus segment, a streaming cache is used to improve the performance of certain types of DMA access, and an IOMMU is provided for translating PCI DMA addresses to physical addresses.

Figure 1-1 shows a block diagram for the PCI leaf. The paths shown are not actual datapaths, but merely the main paths of communications between sub-blocks (not all paths are shown).

1.6.2.2 Block Diagram

Figure 1-1 PCI Leaf Top Level Block Diagram



1.6.2.3 Sub-Block Descriptions

1.6.2.3.1 Interface Controller Block

The IFC, or interface controller, controls the PCI leaf's communication with the internal interface to the Safari block. This is almost entirely new logic for Schizo. The Interface Controller (IFC) is responsible for controlling all the transactions between the other blocks in the PCI leaf (STC, PBM, MDU, IOMMU, etc.) and the asynchronous packet fifos to/from the Safari interface.

This includes:

Receiving, decoding and issuing PIO requests

Returning PIO read data and PIO write completion responses

Arbitrating amongst DMA requests

Forwarding DMA and interrupt requests, and keeping track of outstanding requests

Forwarding DMA write data

Receiving DMA read replies, and routing them to the appropriate block.

Tracking DMA write replies for retiring completed transactions and proper flow control

Forwarding interrupt replies (acks/nacks) to MDU block

handling PIO reads and writes, DMA reads and writes, MDU interrupts, acks and nacks.

1.6.2.3.2 *PCI Bus Module*

The PBM, or PCI Bus Module, is the block directly responsible for operating on the PCI bus as a master and slave device. It operates as a host-PCI bridge, and provides a number of central resources for the PCI bus that it "controls."

The PBM's main features are;

Compliant with PCI Local Bus Specifcation, Revision 2.1.

Supports 33MHz and 66MHz PCI operation

Supports 64-bit DMA and PIO data operations

Supports 64-bit addressing as a target

Dual 64-byte DMA write buffers, quad 64-byte DMA Read buffer, a dual 64-byte PIO write buffer, and a dual 64-byte PIO read buffer

As a PCI central resource, provides arbitration for up to 6 master devices.

As a PCI central resource, provides mechanism for generating PCI Configuration cycles and PCI Special cycles

1.6.2.3.3 *Streaming Cache*

The streaming cache performs three primary functions. The first is to accumulate sequentially addressed PCI write bursts into quantities the size of a system block. The second function is to speculatively prefetch the next (increasing) sequential block(s) of memory for an active PCI read stream. The third function is to act as a local cache for small PCI read accesses to the same block.

The implementation of the streaming cache features:

A fully associative pool of 16 entries shared among read and write streams.

Each entry has storage for two 64-byte blocks of data.

Both reads and writes can ping-pong between lines, and stay connected for large PCI burst sizes.

Dual ported data RAM for concurrent write (flush) and read (fill) operations.

64-bit wide datapaths

Least Recently Used entry allocation scheme

Virtual address tags for low lookup latency.

Physical address page translation for each entry to avoid IOMMU traffic.

One entry allotment per virtual page to reduce the problem of individual misbehaved devices from thrashing the cache.

Individual byte write enables to support PCI bus byte granularity.

Only accesses to virtual pages that are designated by software as streamable pages can use the streaming cache's functions. The cache is not considered to be part of the coherency domain, therefore software intervention is required to ensure a consistent memory image. PCI devices, however, will see program order functionality without any software intervention; reads following writes to the same address will see correct data.

1.6.2.3.4 IO Memory Management Unit

The IOMMU performs virtual to physical address translation during DVMA cycles. PCI master devices provide 32-bit virtual address at the beginning of a DVMA transfer. The IOMMU translates it into 43 bits of physical address plus a cacheable/non-cacheable identifier.

Translations are performed via a 16-entry fully-associative TLB (Translation Lookaside Buffers) implemented in hardware and a TSB (Translation Storage Buffer) which is a software managed data structure (one-level) in memory. The IOMMU performs a TSB lookup (also known as hardware tablewalk) when the translation cannot be found in the TLB, which stores recently used translation information. An error is returned to the PCI master device if the TLB and TSB lookups fails to locate a valid mapping.

The Schizo IOMMU supports two different page sizes, 8K and 64K. Mixed page sizes can be used in the system, but the TSB table lookup only assumes the smaller page size. No overlapping of pages is allowed. Bypass operation is supported to allow devices having their own translation facility to bypass IOMMU.

Translations may be flushed from the TLB via PIO operations. They can be flushed either singly, by their virtual page-number, or in groups, by their context number (the context number is an 8-bit field stored for each translation, which allows related translations to be grouped).

1.6.2.3.5 Mondo Dispatch Unit

The MDU, or Mondo Dispatch Unit, is responsible for generating interrupt packets (Mondo vectors). It has a seven-bit interface to an external interrupt concentrator (RISCx), as well as several internal interrupt lines (timers, ECC logic, PBM errors, etc.). The interface is shared between both MDU blocks (one in each PCI leaf).

When interrupts have been received, they are prioritized by the MDU, and one is dispatched to the Safari bus via the internal interface. The MDU then waits for an ACK or NACK for the interrupt. If the interrupt is NACK'd, the MDU will resend the interrupt after a programmable interval. Only a single interrupt packet is handled by the MDU at any given time.

Before dispatching any PCI related interrupt, the MDU block handshakes with the PBM block to ensure that any outstanding DMA write data will be in memory before the interrupt is received.

1.6.2.3.6 *Performance Monitor*

The PRF block is a performance monitor. It contains two 32-bit counters which can each be programmed to count one of several different event types (such as interrupts, TLB misses, words transferred, etc.).

Address Spaces and Translations 2

This chapter documents the regions of physical address space which Schizo can generate transactions for, or will respond to on each of its interfaces (Safari, UPA, PCI-A, and PCI-B). It also documents the modifications made to addresses as they pass from one interface to another.

All Schizo operations involve at least one transaction on the Safari bus. Transactions which logically transfer data between two leaf blocks do so by way of the Safari bus, so they can conceptually be broken up into separate Leaf⇒Safari and Safari⇒Leaf transactions which are documented below. Transactions in which Schizo does not participate (e.g. peer-to-peer transfers between two devices on the same PCI bus) are not documented here.

2.1 Safari => Leaf interfaces

2.1.1 Safari Cacheable Space

Schizo does not implement any memory in cacheable space, and will not assert Mapped for any Safari cacheable transaction. Schizo does have a small 8-entry cache (Merge Buffer), so it does respond to cacheable transactions by asserting Owned when necessary.

2.1.2 Safari Non-cacheable Space

Schizo decodes and responds to up to 9 separate regions in Safari non-cacheable space. The first region is an 8Mb region that is allocated to every Safari agent. It is located at physical address (0x80.0000*AID) in non-cacheable space, where AID is Schizo's Safari Agent ID. In addition, there are two programmable regions each for UPA, PCI-A, and PCI-B. The base address and actual size for these regions are programmable via registers in the Safari Interface Block. The regions are summarized in Table 2-1. The

Base Address IDs from the table are used later to refer to these regions. BE/LE indicates whether the regions use a Big Endian convention or Little Endian convention for byte ordering.

Table 2-1 Schizo decoding of Safari non-cacheable space

Base Address ID	Usage	BE/ LE	Region Size	Supported Safari Transactions	Generated Leaf Transactions
SafariBase	Maps internal registers for all Schizo leaf blocks.	BE	8 Mb	See Table 2-2	
Upa0Base	Maps UPA64S Slot 0 slave space	BE	2-8 Gb	RIO	UPA P_NCRD_REQ
				RBIO	UPA P_NCBRD_REQ
				WIO	UPA P_NCWR_REQ
				WBIO	UPA P_NCBWR_REQ
Upa1Base	Maps UPA64S Slot 1 slave space	BE	2-8 Gb	RIO	UPA P_NCRD_REQ
				RBIO	UPA P_NCBRD_REQ
				WIO	UPA P_NCWR_REQ
				WBIO	UPA P_NCBWR_REQ
PCI-A_MemBase	Maps PCI-A Memory Space	LE	2-4 Gb	RIO	PCI Memory Read
				RBIO	PCI Memory Read
				WIO	PCI Memory Write
				WBIO	PCI Memory Write
PCI-A_CfgIOBase	Maps PCI-A Configuration Space and PCI-A I/O Space	LE	32 Mb	See Table 2-2	
PCI-B_MemBase	Maps PCI-B Memory Space	LE	2-4 Gb	RIO	PCI Memory Read
				RBIO	PCI Memory Read
				WIO	PCI Memory Write
				WBIO	PCI Memory Write
PCI-B_CfgIOBase	Maps PCI-B Configuration Space and PCI-B I/O Space	LE	32 Mb	See Table 2-2	

Some of these regions are further divided into distinct sections with specific meanings (for example, SafariBase is divided into multiple sections so that each leaf block has space for mapping its internal registers). These subdivisions are summarized in Table 2-2. Again, the section IDs from the table will be used later to refer to particular sections of within Schizo's address regions.

Table 2-2 Subdivision of Schizo Address Regions

Section ID	Usage	BE/ LE	Base Address	Size	Supported Safari Transactions	Generated Leaf Transaction
FCode	Safari device ID register (Schizo does not support optional FCode prom).	BE	SafariBase+ 0x000000	4 Mb	RIO (8 byte, aligned)	Internal register access
					WIO (8 byte, aligned)	

Table 2-2 Subdivision of Schizo Address Regions

Section ID	Usage	BE/ LE	Base Address	Size	Supported Safari Transactions	Generated Leaf Transaction
SafariCSRBase	Safari and UPA interface blocks internal registers	BE	SafariBase+ 0x400000	1 Mb	RIO (8 byte, aligned)	Internal register access
					WIO (8 byte, aligned)	
PCI-A_CSRBase	PCI-A leaf internal registers	BE	SafariBase+ 0x600000	1 Mb	RIO (8 byte, aligned)	Internal register access
					WIO (8 byte, aligned)	
PCI-B_CSRBase	PCI-B leaf internal registers	BE	SafariBase+ 0x700000	1 Mb	RIO (8 byte, aligned)	Internal register access
					WIO (8 byte, aligned)	
PCI-A_ConfigBase	Maps PCI-A Configuration Space	LE	PCI-A_CfgIOBase+ 0x0000000	16 Mb	RIO (0-4 bytes, spanning single word only)	PCI Configuration Read
					WIO (0-4 bytes, spanning single word only)	PCI Configuration Write
PCI-A_IOBase	Maps PCI-A I/O Space	LE	PCI-A_CfgIOBase+ 0x1000000	16 Mb	RIO (0-4 bytes, spanning single word only)	PCI I/O Read
					WIO (0-4 bytes, spanning single word only)	PCI I/O Write
PCI-B_ConfigBase	Maps PCI-B Configuration Space	LE	PCI-B_CfgIOBase+ 0x0000000	16 Mb	RIO (0-4 bytes, spanning single word only)	PCI Configuration Read
					WIO (0-4 bytes, spanning single word only)	PCI Configuration Write
PCI-B_IOBase	Maps PCI-B I/O Space	LE	PCI-B_CfgIOBase+ 0x1000000	16 Mb	RIO (0-4 bytes, spanning single word only)	PCI I/O Read
					WIO (0-4 bytes, spanning single word only)	PCI I/O Write

The Safari physical address for any given Safari non-cacheable transaction to which Schizo will respond can thus be divided as BaseAddress + Offset, where BaseAddress is one of the above regions or sections, and Offset is less than the size of the corresponding region/section. Software must guarantee that the regions do not overlap so that this division is unique for a given address. Schizo's behavior is undefined if the regions do overlap. As the physical address is passed to the leaf block, it typically undergoes some transformation, which is documented below for each different destination.

2.1.2.1 *Accesses to UPA*

- UPA<38:33> = 000000
- UPA<32:4> = Offset<32:4>

2.1.2.2 Accesses to PCI

2.1.2.2.1 PCI Configuration Space

When accessing PCI Configuration Space, Offset<23:16> defines the PCI Bus Number that is being addressed, Offset<15:11> defines the Device Number, Offset<10:8> defines the Function Number, and Offset<7:2> defines the Register Number.

If the Bus Number for a configuration access matches the Bus Number of the PCI Leaf being accessed, a Type 0 Configuration Cycle will be generated. If the Bus Number is greater than the PCI Leaf's Bus Number, but less than or equal to the PCI Leaf's Subordinate Bus Number, a Type 1 Configuration Cycle will be generated. Otherwise, the transaction will cause an error.

For Type 0 Configuration Cycles, the PCI address that is issued is built as follows:

- PCI<31:11> = 2 ^ DeviceNumber[i.e. PCI<11+DeviceNumber> will be set to 1, all other bits in PCI<31:11> will be set to 0].
- PCI<10:8> = Function Number
- PCI<7:2> = Register Number
- PCI<1:0> = 00

In addition, if DeviceNumber falls within the supported range (1-4 for PCI-A, and 1-6 for PCI-B), one of Schizo's IDSEL outputs will be asserted as follows:

- IDSEL<n:0> = 2 ^ (DeviceNumber-1) [i.e. IDSEL<DeviceNumber-1> will be set to 1, all other bits in IDSEL<n:0> will be set to 0. For PCI-A, n=3, and for PCI-B, n=5].

Example: if Offset<23:0> = 0x121C00, the configuration access is to Bus Number 0x12, Device Number 3, Function Number 4. If the Bus Number of the PCI Leaf is 0x12, a Type 0 Configuration Cycle will be generated with PCI<31:0> = 0x00004400, and IDSEL<n:0> = 0x04.

A PCI Special Cycle is simply a special case of a Type 0 Configuration Cycle, where Offset<15:2> = 0x3fc0.

For Type 1 Configuration Cycles, the PCI address that is issued is built as follows:

- PCI<31:24> = 00000000
- PCI<23:2> = Offset<23:2>
- PCI<1:0> = 01

See the PCI Specification for more details on the format of Configuration Space addresses.

2.1.2.2.2 PCI IO Space

- PCI<31:24> = 00000000
- PCI<23:2> = Offset<23:2>
- PCI<1:0> = 00

2.1.2.2.3 PCI Memory Space

- PCI<31:2> = Offset<31:2>
- PCI<1:0> = 00

2.1.3 Safari Interrupt Space

Schizo does not respond to any Safari Interrupt transactions.

2.1.4 Safari Admin Space

Schizo does not respond to any Safari Admin transactions.

2.2 UPA => Safari

The UPA64S is a slave only interface, so there are no transactions for Schizo to respond to on this bus.

2.3 PCI => Safari

PCI-A and PCI-B are identical in terms of how the PCI address spaces are handled; this section applies to both.

2.3.1 PCI Configuration Space

Schizo does not respond to any Configuration Read or Configuration Write cycles on the PCI bus. Schizo is the central resource for each of its PCI buses, and is expected to be the only device generating configuration cycles. Accesses from the Safari bus that target configuration registers that are internal to a PCI Leaf Block will be serviced by the PCI Leaf without generating a configuration cycle on the PCI bus.

2.3.2 PCI IO Space

Schizo does not respond to any PCI IO Space transactions (IO Read or IO Write command types).

2.3.3 PCI Memory Space

This is the space in which DVMA, and DMA (IOMMU bypass) activity take place. PCI peer-to-peer activity may also use this address space, and is included in the tables below, although Schizo is not involved in these transfers.

2.3.3.1 PCI Address Translation Modes

The final destination and address translation of a PCI Memory transaction is based on:

- PCI addressing mode used: 64-bit (DAC) vs. 32-bit (SAC)
- PCI address bit <31> in SAC mode
- Value of MMU_EN in the IOMMU Control Register
- Value of PCI address bits <63:50> in DAC mode

In all cases, Schizo will only support bursts as a target device in Linear Incrementing mode (i.e. PCI<1:0> must be 00). If any of the reserved modes are used, Schizo will only transfer a data phase, and then issue a target disconnect.

The following table shows the various ways that Schizo as a PCI target device deals with PCI Memory Space addresses.

Table 2-3 PCI DVMA Modes of Operation

Mode	Addr<31>	MMU_EN	Addr<63:50>	Result
SAC	0	X	N/A	PCI peer-to-peer (Ignored by Schizo)
SAC	1	0	N/A	Pass-through
SAC	1	1	N/A	IOMMU Translation (DVMA)
DAC	X	X	0x0000-0x3FFE	Ignored by Schizo
DAC	X	X	0x3FFF	Bypass (DMA)

2.3.3.1.1 *Pass-through*

In pass-through mode, Safari<42:31> = 0x000, Safari<30:4> = PCI<30:4>. Pass-through transfers always generate cacheable transactions on Safari, and are always done in consistent mode.

2.3.3.1.2 *IOMMU Translation mode*

In IOMMU translation mode, the physical address is obtained by performing a virtual to physical translation through the IOMMU. The value of the cacheable (C) bit in the TTE for the virtual page determines whether the Safari transaction generated is cacheable or non-cacheable. The value of the streamable (S) bit in the TTE determines whether the transaction used consistent or streaming mode.

2.3.3.1.3 *PCI peer-to-peer mode*

In peer-to-peer mode, two devices on the same PCI bus transfer data without any involvement from Schizo. The master device simply puts out the PCI address to which the target device has been mapped. Whether there is any subsequent address translation involved is device dependent. If no device has been mapped at the target address, the PCI master device will terminate its cycle with a Master-Abort.

2.3.3.1.4 *Bypass mode*

In bypass mode, the Safari<42:4> = PCI<42:4>. A cacheable Safari transaction will be generated if PCI<42> = 0, otherwise a non-cacheable transaction type will be used. Bypass mode transactions are always done in consistent mode.

2.3.3.2 *Safari Transactions Generated*

The exact Safari transaction(s) initially generated for a given PCI DMA transaction depends upon:

- Transfer direction: read/write
- Cacheability (C), as documented above.
- Transfer size.

The transfer size can be a complete cache line (64 bytes), or a partial cache line, in which case the size refers not to the number of valid bytes, but to the number of bytes spanned by the valid bytes, and then extended to the nearest 8-byte aligned boundary on either side (e.g a transfer in which only bytes 31 and 32 are valid has a 16-byte transfer size)

In consistent mode, the transfer size is determined solely by the burst length on the PCI bus and the byte enables of each data phase. Schizo does not allow a consistent mode transfer to cross a 64-byte aligned boundaries, and will disconnect on the PCI if this is attempted.

In streaming mode (which is always cacheable), the transfer size is determined by the circumstances that cause the streaming cache to issue a Safari request, and by the recent history of the streaming cache line.

Note that this table does not show all of the potential Safari transactions that may be issued in servicing a PCI DMA request. Extra transactions may be generated due to SSM intervention.

Table 2-4 Safari Transactions generated for PCI DMA activity

C	R/W	Size (bytes)	Safari Result(s)
0	Read	Any	ReadStream
0	Write	64	WriteStream
0	Write	Partial 0-64	ReadToOwn WriteBack
1	Read	Any	ReadBlockIO
1	Write	64	WriteBlockIO
1	Write	Partial 0	WriteIO
1	Write	Partial 8-64	n WriteIO transactions, where n = Size / 8

2.3.4 Interrupts

For any interrupts generated by the PCI leaf (this includes all interrupts that Schizo is currently capable of generating), a Safari INT transaction will be issued by Schizo, using the following Safari address:

- Safari<42:39> = 0000
- Safari<38:34> = Schizo's Node ID (NId field of Schizo Control/Status Register)
- Safari<33:29> = Schizo's Agent ID (programmed via Schizo's Z_AID pins, readable in AID field of Schizo Control/Status Register)
- Safari<28:24> = 00000

- Safari<23:19> = Target Node ID (T_NodeID field of Interrupt Mapping Register for interrupt being generated)
- Safari<18:14> = Target Agent ID (T_AgentID field of Interrupt Mapping Register for interrupt being generated)
- Safari<13:4> = 0000000000

Little-endian Address Spaces 3

3.1 Overview

The main interface of Schizo, the Safari bus, is big-endian. Other interfaces, notably the two PCI buses, are little-endian. Schizo provides the necessary support to connect the two together in a consistent fashion. The main feature is called “byte twisting”: from a hardware perspective, the bytes on the datapath from a PCI bus are twisted around before connecting to any other datapath within Schizo, so that bits 63:56 map to bits 7:0, bits 55:48 map to bits 15:8, etc. From another perspective, this ensures that logical byte lanes are connected: the byte at address 0 on the big-endian side is directly wired to the byte at address 0 on the little-endian side. As a result, all byte-sized PIOs and byte-stream DMA is handled correctly. This, along with other features built into SPARC V9 processors, provides a mechanism for all PIO and DMA activity to/from the PCI bus to take place correctly.

3.2 Big- and Little-endian regions

3.2.1 Address Space

Schizo responds to several different regions of non-cacheable address space on the Safari bus, as documented in Chapter , “Address Spaces and Translations.” As indicated there, the following regions are little-endian, and all accesses to them use byte twisting.

- PCI-A_MemBase
- PCI-A_ConfigBase
- PCI-A_IOBase
- PCI-B_MemBase
- PCI-B_ConfigBase
- PCI-B_IOBase

All other address regions are big-endian, and there is no byte twisting done for accesses other than for the regions listed above.

3.2.2 Internal blocks

Given the above breakdown of address regions, most of Schizo's internal blocks are in big-endian address space: the 64-bit data paths within the Schizo design blocks are connected to the Schizo's Safari data bus with no byte twisting. The exceptions are the PBM blocks within each PCI leaf. Since the PBM controls the little-endian PCI bus, it is considered to be a little-endian block. For each data interface entering/leaving a PBM block, byte twisting is in effect (in addition to interfacing to PIO and DMA buses within the PCI leaf, each PBM has data interfaces with its associated streaming cache block).

Note – Each PBM also contains some internal control/status registers mapped into big-endian spaces (PCI-A_CSRBase and PCI-B_CSRBase). So that these registers are not affected by the byte twisting of the PBM's data paths to the rest of the chip, within the PBM, the data path to these registers is "retwisted".

3.3 Byte Twisting

Figure 3-1 diagrams what is meant by byte twisting. It shows how data is manipulated from a 32-bit little-endian PCI bus to a 64-bit big-endian Safari bus. The case of a 64-bit PCI bus is a straightforward modification of this diagram, and won't be shown.

For each bus, a typical connection to memory is shown, along with the byte addresses of the memory. This is mainly for reference - it is one way of showing exactly what is meant by big- or little-endian. It helps to show that the "logical" byte lanes of each bus are correctly connected through Schizo.

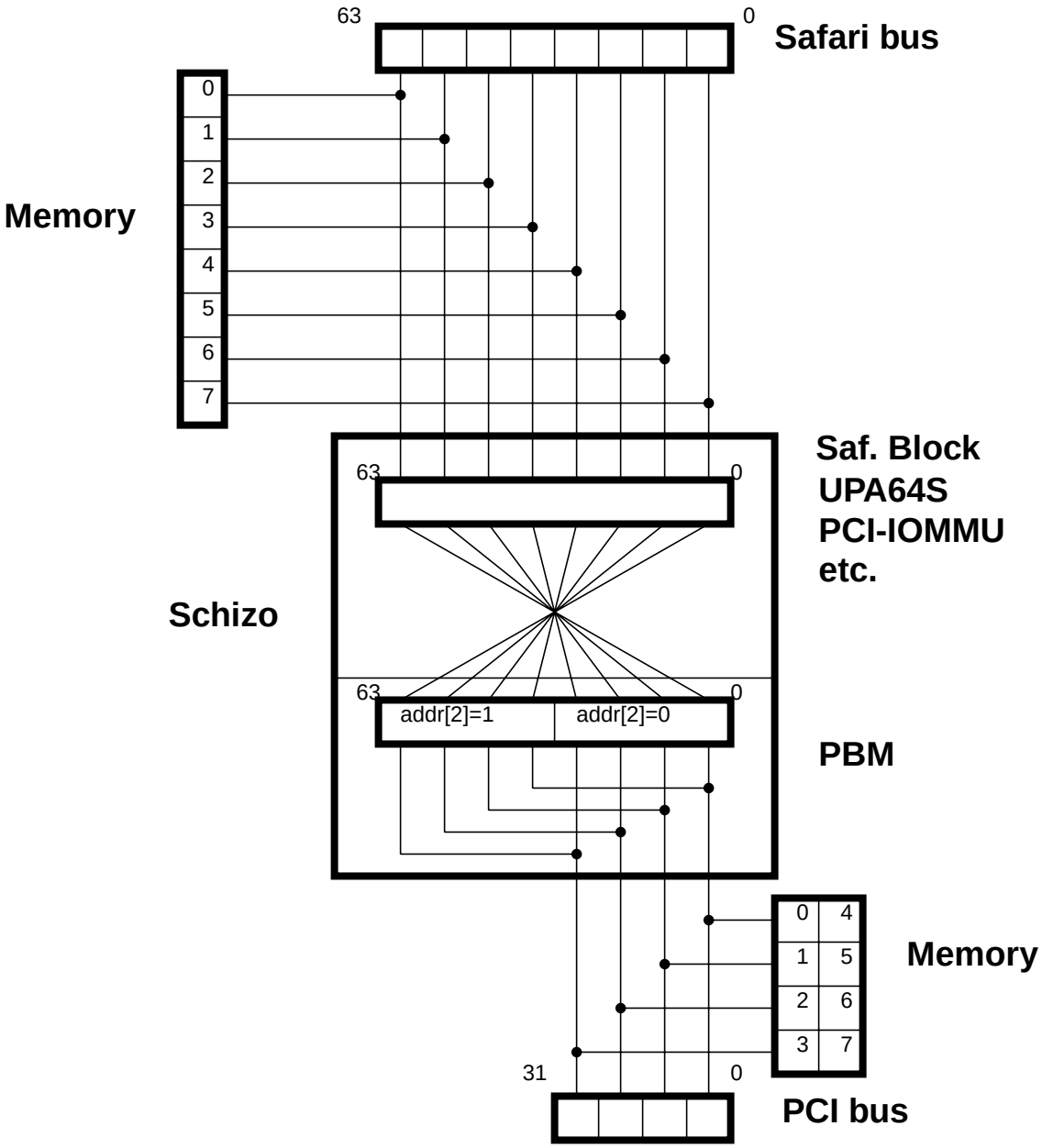


Figure 3-1 Schizo Byte Twisting

3.4 *Specific Cases*

The following sections detail specific types of data transfers, and how correct byte ordering is maintained in each case.

3.4.1 PIOs

3.4.1.0.1 Normal

Due to the byte twisting, all byte sized PIOs work correctly. The byte lane used for a given address on the big-endian side is directly wired to the byte lane used for that address on the little-endian side.

For any access larger than a byte, byte-twisting is not sufficient. For example, if the 32-bit value 0x12345678 is written to a 32-bit register on a PCI device, the PCI device will interpret the value as 0x78563412 after byte twisting.

To correct for this, SPARC V9 CPUs have special support for little-endian access. By either marking the page containing the PCI register as little-endian in the processor's MMU, or by using one of the little-endian Address Space Identifiers (ASIs), the CPU will alter its ordering of the bytes, in effect undoing the byte-twisting that will be done by Schizo, so that the PCI device correctly sees 0x12345678.

3.4.2 DMA

3.4.2.0.1 Data streams

Because byte lanes at the same address are connected, DMA of byte streams works correctly without any further intervention. A PCI device that receives the byte stream 0x01 0x02 0x03 0x04 would pack the bytes into a 32-bit register starting with the LSB of the register, i.e. 0x04030201. If this were transferred to memory on the PCI bus, the value 0x01 would be at the lowest memory location, as desired.

After byte twisting, the value that appears on the Safari bus would be 0x01020304. Since the MSB on the Safari bus is the lowest memory location, the value 0x01 is still stored at the lowest memory location, as desired.

3.4.2.0.2 Descriptors

This case is similar to PIOs of size greater than one byte. With just byte twisting, a DMA descriptor access would get the wrong byte ordering. For example, if the value 0x12345678 were set up in an address field in a descriptor, a PCI device using DMA to fetch the descriptor would see the value as 0x78563412 instead.

To avoid this, the little-endian features of the processor are used again. Processor loads and stores to the descriptors should be specified as little-endian. This will re-order the bytes in memory when the descriptor is built so that after byte twisting, the PCI device sees the correct value.

Register Descriptions 4

4.1 General Information

The following notes, warnings and restrictions apply to the registers throughout all parts of Schizo.

4.1.1 Abbreviations Used

The following abbreviations are used throughout this chapter:

- R - Read only: a register field that is not writable
- R/W - Read/write: A standard register field
- R/W1C - Read / write 1 to clear: Writing a 0 to bits in this field has no affect, but writing a 1 to a bit in this field will cause that bit to be set to 0.
- W - Write only: reads of this field return undefined data
- DMA - Direct Memory Access: A transaction initiated by a leaf block resulting in a Safari transaction
- DVMA - Direct Virtual Memory Access: A subclass of DMA transactions in which the address of the leaf transaction is treated as a virtual address and translated by an MMU.

4.1.2 Access Size

Each register in Schizo has a natural access size, which is documented below. For most registers this size is 8 bytes, but there are exceptions. Register accesses should always be done with the indicated access size, or undefined behavior may result, including, but not limited to:

- Incorrectly sized writes may still write data to the entire register
- Incorrectly sized reads may not trigger a side-effect (if any) of the register
- Reads or writes that are too large (span multiple registers) can return incorrect data or corrupt any of the addressed registers.

4.1.3 Unimplemented Addresses

Any address within Schizo's register address space that is not documented here as belonging to a specific register is reserved and should not be read or written by software. A read will return undefined data, and a write can corrupt the contents of another register within Schizo, as Schizo may only decode as few address bits as needed to distinguish existing registers.

4.1.4 Physical Addresses

Complete Safari physical addresses are not shown in here for any registers, since each address region, except SafariBase, that Schizo responds to is directly relocatable. The physical address for SafariBase is:

```
{10'h200, NID, AID, 23'd0}
```

where NID is Schizo's Safari Node ID and AID is Schizo's Safari Agent ID, (from the Schizo Control/Status Register.)

Instead, all register addresses are documented as offsets within a particular address region. Unless otherwise indicated, all offsets are with respect to the SafariBase region (this accounts for almost all of Schizo's registers).

4.2 Safari Interface

Table 4-1 Safari Register Offsets

Register	Offset	Access Size
Safari Device ID Register	0x00.0000	8 bytes
UPA0 Address Match Register	0x40.0000	8 bytes
UPA0 Address Mask Register	0x40.0008	8 bytes
UPA1 Address Match Register	0x40.0010	8 bytes
UPA1 Address Mask Register	0x40.0018	8 bytes
PCI-A_Mem Address Match Register	0x40.0040	8 bytes
PCI-A_Mem Address Mask Register	0x40.0048	8 bytes
PCI-A_Cfg_IO Address Match Register	0x40.0050	8 bytes
PCI-A_Cfg_IO Address Mask Register	0x40.0058	8 bytes
PCI-B_Mem Address Match Register	0x40.0060	8 bytes
PCI-B_Mem Address Mask Register	0x40.0068	8 bytes
PCI-B_Cfg_IO Address Match Register	0x40.0070	8 bytes
PCI-B_Cfg_IO Address Mask Register	0x40.0078	8 bytes
Schizo Control/Status Register	0x41.0000	8 bytes
Safari Error Control Register	0x41.0008	8 bytes
Safari Interrupt Control Register	0x41.0010	8 bytes
Safari Error Log Register	0x41.0018	8 bytes
ECC Control Register	0x41.0020	8 bytes
UE AFSR	0x41.0030	8 bytes
UE AFAR	0x41.0038	8 bytes
CE AFSR	0x41.0040	8 bytes
CE AFAR	0x41.0048	8 bytes
Safari Energy Star Control Register	0x41.0050	8 bytes
Safari Soft Pause Register	0x41.0058	8 bytes
Schizo Queue Control Register	0x41.1000	8 bytes
Safari DTag Diagnostic Registers	0x41.2000 - 0x41.2070	8 bytes
Safari CTag Diagnostic Registers	0x41.3000 - 0x41.3070	8 bytes
Safari Debug Registers	0x41.4000 - 0x41.4018	8 bytes
Safari Performance Control Register	0x41.7000	8 bytes
Safari Performance Counter Register	0x41.7008	8 bytes

4.2.1 Safari Device ID Register

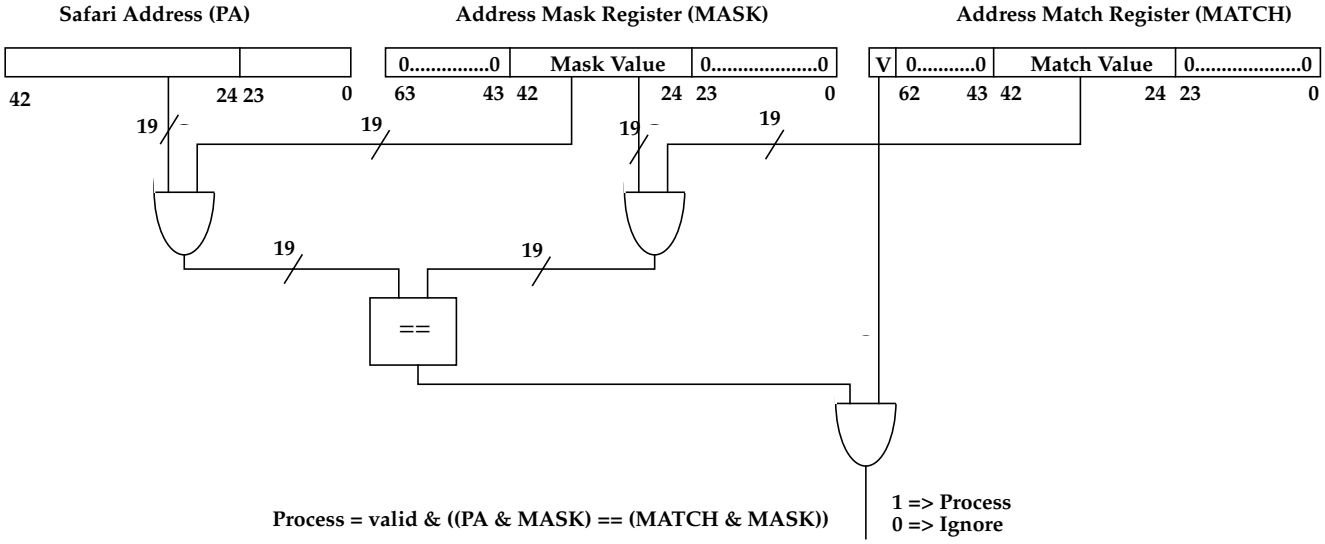
Table 4-2 Safari Device ID Register

Bits	Field	Description	Default Value	Type
63:56	Cookie	This field is provided so that the open boot PROM code detects that Schizo does not support an Fcode PROM.	0xFC	R
55:27	Rsvd	Reserved. Read as zero		R
26:22	NID	Safari Node ID. The Node ID is read-only here, but may be set via JTAG or through the Schizo Control and Status register.	0x00	R
21:17	AID	Safari Agent ID. The Agent ID is read-only here, but is set during reset from external pins on Schizo, and may also be set via JTAG.	-	R
16	M/S	Master/Slave. Schizo is both a master and slave device on Safari	1	R
15:10	MID	Manufacturer ID. (SCHIZO)= 0x2A, (ELE) = 0x2B.	0x2A	R
9:4	MT	Module Type.	0x15	R
3:0	MR	Module Revision. The following values of MR are currently in use: 0x0 - Schizo 1.0 netlist (Schizo 1.0 parts) 0x2 - Schizo 2.0 netlist (Schizo 2.0 parts) 0x3 - Schizo 2.1 netlist (Schizo 2.1 parts) 0x4 - Schizo 2.2 netlist (Schizo 2.2 parts) 0x5 - Schizo 2.3 netlist (Schizo 2.3 parts, ELE 1.0 parts) 0x6 - Schizo 2.4 netlist (Schizo 2.4 parts) Latest rev: 0x7 - Schizo 2.5 netlist (Schizo 2.5 parts, ELE 1.1 parts)	0x7	R

4.2.2 Address Match and Mask Registers

Schizo is using a common implementation for all of its address match and mask registers although in some cases this leads to extra bits that are not strictly necessary as they allow Schizo to map a region larger than what is implemented by the corresponding I/O bus (UPA64S or PCI).

All incoming Safari addresses are checked by the Safari interface block against the values programmed in all of the Address Match and Mask registers to decide if Schizo is the target of the transaction. The address filtering is performed as described by the picture below:



Note – Note both Mask and Match values should be programmed before setting Valid. This can be done by writing the Mask register and then the Match register (with a Match value and Valid.)

Figure 4-1 Address Filtering in Schizo

When a Safari request is filtered positively it is forwarded to the appropriate block (Safari registers, UPA, PCI A or PCI B).

The format of the Address Match registers is:

Table 4-3 Address Match Register

Bits	Field	Description	Reset Value	Type
63	V	Valid. Mapping is enabled when set to 1. Disabled when set to 0	0	R/W
62:43	Rsvd	Reserved. Read as zero.	0x0 0000	R
42:24	Match	Match Value. This is the value of the base address to be matched against.	Undefined	R/W
23:0	Rsvd	Reserved. Read as zero.	0x00 0000	R

The format of the Address Mask registers is:

Table 4-4 Address Mask Register

Bits	Field	Description	Reset Value	Type
63:43	RESERVED	Reserved, read as 0.	0x00.0000	R
42:36	MASK_HI	High order mask bits, always set to 1s.	0x7F	R
35:24	Mask	Mask Value. This mask is used to remove the least significant bits before matching. It defines the size of the mapped region. It must be string of 1 followed by a string of 0.	Undefined	R/W
23:0	Rsvd	Reserved. Read as zero	0x00 0000	R

Each pair of Address Match and Mask register can map a space between 16 MB and 64 GB, although not all possible sizes make sense. Table 4-5 lists the maximum, minimum and recommended sizes for each address space. Internally, Schizo will ignore any

Table 4-5 Address Match/Mask size limitations

Address Space	Min Size	Max Size	Recommended		Comments
			Size	Mask[35:24]	
UPA0	2GB	8GB	8GB	0xE00	Existing UPA framebuffers require 2GB, future devices may require up to 8GB. UPA does not support >8GB per port.
UPA1					
PCI-A_Mem	16MB	4GB	2GB	0xF80	Limits amount of slave space available for mapping PCI devices. PCI doesn't support more than 4GB, while Schizo normally limits slave space to 2GB at most. 4GB should only be used for diagnostic loopback mode.
PCI-B_Mem					
PCI-A_Cfg_IO	32MB	32MB	32MB	0xFFE	If set to 16MB, either PCI Config transactions or PCI IO transactions will be disabled (determined by value of Match[24]).
PCI-B_Cfg_IO					

address bits beyond the maximum size listed here. If programmed to a larger size, the given address space will be mirrored as many times as necessary in the Safari physical address space.

UPA0 and UPA1 spaces are also qualified by slot empty and soft reset, so that MappedOut is not asserted when a device is not able to respond. (See UPA CSR definitions.)

4.2.3 Schizo Control/Status Register

Table 4-6 Schizo Control/Status Register

Bits	Field	Description	Reset Value	Type
63:58	Rsvd	Reserved. Read as zero.	0x00	R
57:34	DTL Mode	DTL mode for Driver/Receiver groups. Each pair of bits represents the operating mode for an independent group of DTL I/Os in Schizo. Group 0 is bits [35:34]. The groups are defined in Table 4-7 (see Safari Electrical Spec for more details). Encodings are: 0x0 - Reserved 0x1 - DTL-end mode 0x2 - DTL-mid mode 0x3 - DTL-2 mode This field is read-only here. The DTL modes are set upon reset from a table indexed by the value of the Z_SYSCODE inputs to Schizo. The DTL modes are JTAG Shadow R/W.	-	R
33:32	STO	Safari Time-Out interval. Values are: • 0x0 => ∞ (Safari timeouts disabled) • 0x1 => 2^{28} Safari clock cycles (normal) • 0x2 => 2^{15} Safari clock cycles (debug) • 0x3 => 2^{9} Safari clock cycles (simulation) This controls the duration of each Safari timeout event defined in Table 4-11. All timeouts are imprecise, and fall into the following range: STO <= Actual Timeout <= 2 * STO See cautions on changing STO while any timeout events are enabled below. The STO value is JTAG Shadow R/W.	0x0	R/W
31:30	Rsvd	Reserved. Read as zero.	0x0	R
29:25	NID	Safari Node ID. See warning below. The NID value is JTAG Shadow R/W.	0x00	R/W
24:20	AID	Safari Agent ID. The Agent ID is read-only here, but is set during reset from external pins on Schizo. The AID value is JTAG Shadow R/W.	-	R
19:15	Rsvd	Reserved. Read as zero.	0x00	R
14:9	LPA_BND	Local Physical Address Bound.	0x3f	R/W
8:3	LPA_BASE	Local Physical Address Base. The Local Physical Address Base and Bound are used to define the Local Physical Address range for COMA SSM system. It defines addresses that are in COMA space. A Safari address is decoded as “local” to a node if LPA_BASE ≤ PA[42:37] < LPA_BND.	0x00	R/W

Table 4-6 Schizo Control/Status Register

Bits	Field	Description	Reset Value	Type
2	SLOWSNP	Slow snoop. Controls SNOOP_DELAY parameter from Safari spec: SLOWSNP=0 => SNOOP_DELAY=0 SLOWSNP=1 => SNOOP_DELAY=2 Value is set upon reset from a table indexed by the value of the Z_SYSCODE inputs to Schizo (see Table 4-7). Also JTAG Shadow R/W.	-	R
1	HBM	Hierarchical Bus mode (ADDR_REP in Safari spec). HBM=0 => No address repeaters in system HBM=1 => System has address repeaters Value is set upon reset from a table indexed by the value of the Z_SYSCODE inputs to Schizo (see Table 4-7). Also JTAG Shadow R/W.	-	R
0	SSM	Scalable Shared Memory mode enable. This bit may not be changed while DMA is enabled. Also JTAG Shadow R/W.	0	R/W

4.2.3.0.1 *Safari Time-Out*

Timeouts may be changed between “disabled” (0x0) and any “enabled” state without causing spurious timeouts. When in the “disabled” state timeouts are frozen, the interval is literally infinite.

Warning – Changing the timeout interval between “enabled” states may cause spurious timeouts. Changing the timeout interval from one “enabled” state, to the “disabled” state, and then to a different “enabled” state may timeout in the shorter of the two intervals.

4.2.3.0.2 *Schizo NodeID*

Warning – Changing the NID field in the Schizo Control/Status register causes the location of Schizo’s registers to move in physical address space. The timing of this change is imprecise. A subsequent access using the old address may still work, while a subsequent access with the new address may fail.

The following sequence may be used to update the NID field (assuming that loads are blocking):

- store @old_CSR_address, new_NID
- load @old_CSR_address (this load may receive an unmapped error)
- load @new_CSR_address (this load should work unless a real error occurs)

Because of the delays associated with writing to a Schizo register, the potential error in the above sequence will typically not occur, although it is possible under some conditions.

4.2.3.0.3 *Schizo Syscode Table*

Schizo has an on-chip table which is used to initialize several fields in the Schizo Control/Status register. The table is indexed by the value of the Z_SYSCODE[2:0] pins on Schizo during reset. The values in the table are:

Table 4-7 Schizo Syscode Table

	Excalibur		**Daktari**		**Serengeti**	**Reserved**	**Verif-only**
Z_SYSCODE[2:0]	**5 (MP)**	**7 (UP)**	**3 (Mid)**	**2 (End)**	**0**	**4,1**	**6**
HBM (S_CSR[1])	0	0	1	1	1	-	0
SLOWSNP (S_CSR[2])	0	0	0	0	1	-	1
DTL GROUP 0 (S_CSR[35:34]): • S_COMMAND_L[1:0] • S_ADDRESS_L[42:4] • S_MASK_L[9:0] • S_ATRANSID_L[8:0] • S_ADDRPTY_L	3 (DTL-2)	1 (DTL-end)	2 (DTL-mid)	1 (DTL-end)	1 (DTL-end)	-	2 (DTL-mid)
DTL GROUP 1 (S_CSR[37:36]: • S_INCOMING_L • S_PREREQIN_L	1 (DTL-end)	1 (DTL-end)	2 (DTL-mid)	1 (DTL-end)	1 (DTL-end)	-	3 (DTL-2)
DTL GROUP 2 (S_CSR[39:38]): • S_ADDRARBOUT_L • S_ADDRARBIN_L[4:0]	3 (DTL-2)	1 (DTL-end)	2 (DTL-mid)	1 (DTL-end)	1 (DTL-end)	-	3 (DTL-2)
DTL GROUP 3 (S_CSR[41:40]): • S_PAUSEOUT_L • S_MAPPEDOUT_L • S_SHAREDOUT_L • S_OWNEDOUT_L	3 (DTL-2)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	-	2 (DTL-mid)
DTL GROUP 4 (S_CSR[43:42]): • S_PAUSEIN_L • S_OWNEDIN_L • S_SHAREDIN_L • S_MAPPEDIN_L • Z_PAUSEIN1_L • Z_OWNEDIN1_L • Z_SHAREDIN1_L • Z_MAPPEDIN1_L	1 (DTL-end)	1 (DTL-end)	2 (DTL-mid)	1 (DTL-end)	1 (DTL-end)	-	3 (DTL-2)
DTL GROUP 5 (S_CSR[45:44]): • S_DTRANSID_L[8:0] • S_DTARG_L • S_DSTAT_L[1:0]	3 (DTL-2)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	-	2 (DTL-mid)
DTL GROUP 6 (S_CSR[47:46]): • S_TARGID_L[8:0] • S_TTRANSID_L[8:0]	3 (DTL-2)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	-	2 (DTL-mid)

Table 4-7 Schizo Syscode Table

Z_SYSCODE[2:0]	Excalibur 5 (MP)	Excalibur 7 (UP)	Daktari 3 (Mid)	Daktari 2 (End)	Serengeti 0	Reserved 4,1	Verif-only 6
DTL GROUP 7 (S_CSR[49:48]): • S_DATAARBGNT_L • S_DATAARBREQ_L • S_TARGARBGNT_L • S_TARGARBREQ_L • Z_DATAARBGNT1_L • Z_DATAARBREQ1_L • Z_TARGARBGNT1_L • Z_TARGARBREQ1_L	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	-	3 (DTL-2)
DTL GROUP 8 (S_CSR[51:50]): • S_PARITYSINGLE_L • S_PARITYBIDI_L • Z_PARITYSINGLE1_L • Z_PARITYBIDI1_L	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	-	2 (DTL-mid)
DTL GROUP 9 (S_CSR[53:52]): • Z_DATA_L[71:0] • Z_DPAR_L	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	-	2 (DTL-mid)
DTL GROUP 10 (S_CSR[55:54]): • Z_SCTL#_L[1:0] (# = 0,1,2,3) • Z_ZCTL#_L[2:0] (# = 0,1,2,3)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	1 (DTL-end)	-	2 (DTL-mid)
DTL GROUP 11 (S_CSR[57:56]): • S_ERROR_L • S_FREEZE_L • S_FREEZEACK • S_CHANGE_L	2 (DTL-mid)	2 (DTL-mid)	2 (DTL-mid)	2 (DTL-mid)	1 (DTL-end)	-	3 (DTL-2)

Note – Z_PAUSEIN1_L, Z_OWNEDIN1_L, Z_SHAREDIN1_L, & Z_MAPPEIN1_L need external pullups when Z_SYSCODE = 0x3, i.e. Daktari Mid configurations.

Note – S_ERROR_L, S_FREEZE_L, S_FREEZEACK, & S_CHANGE_L need external pullups when Z_SYSCODE = 0x5 or 0x7, either Excalibur configuration.

Note – S_FREEZEACK has been specially modified to tri-state when “driving high” and configured as “DTL-mid”. (I.e. it isn’t true DTL-mid.)

4.2.4 *Safari Error Control/Log Registers*

The Safari Error Control Register controls which of several possible errors will result in the assertion of the S_ERROR_L signal by Schizo on the Safari bus. The Safari Interrupt Control Register controls which of the same set of possible errors will cause Schizo to issue a Safari Error Interrupt, and the Safari Error Log Register logs which of the errors have been detected.

All three registers share a similar format. The common portion is detailed in Table 4-11.

Table 4-8 Safari Error Control Register

Field	Bits	Reset	Description	Type
ERREN	63	0	Global error report enable. When set to 0, no errors are reported on S_ERROR_L. When set to 1, errors enabled in the rest of this register are reported on S_ERROR_L.	R/W
Individual Error Enables	62:0	-	See Table 4-11 for bit assignments. Reserved bits are read only, all others are R/W. When any of these bits are set to 1, Schizo will assert S_ERROR_L if the associated error is detected and ERREN is also 1. The value of these enables is persistent across reset. Bits are JTAG Shadow R/W.	R/W

Table 4-9 Safari Interrupt Control Register

Field	Bits	Reset	Description	Type
SE_INTEN	63	0	Global error interrupt enable. When set to 0, no errors are reported via a Safari Error Interrupt. When set to 1, errors enabled in the rest of this register are reported as a Safari Error Interrupt.	R/W
Individual Interrupt Enables	62:0	-	See Table 4-11 for bit assignments. Reserved bits are read only, all others are R/W. When any of these bits are set to 1, Schizo will issue a Safari Error Interrupt if the associated error is detected and SE_INTEN is also 1. The value of these enables is persistent across reset. Bits are JTAG Shadow R/W.	R/W

Table 4-10 Safari Error Log Register

Field	Bits	Reset	Description	Type
ERR_OUT	63	-	Error Out Asserted. This bit is set to a 1 anytime Schizo asserts S_ERROR_L. It is persistent across reset, and can only be cleared by writing to this register.	R/W1C
Individual Error Logs	62:0	-	See Table 4-11 for bit assignments. Reserved bits are read only, all others are R/W1C. Schizo will unconditionally set a log bit to 1 when the associated error is detected, regardless of the state of error and interrupt enables. The value of these enables is persistent across reset. Bits are JTAG Shadow R/W.	R/W1C

Table 4-11 Safari common error bit assignments

Error	Bit	Description
Bad_Safari_Cmd	62	Unrecognized Safari command received
SSM_Disabled	61	Safari SSM command received and SSM mode not set
Bad_MIT-A_Cmd	60	Unrecognized command received from internal PCI-A leaf
Bad_MIT-B_Cmd	59	Unrecognized command received from internal PCI-B leaf
(Bad_MIT-N_Cmd)	58	Unused
Reserved	57:14	Reserved, read-only bits, read as 0
CPU1_Parity_Single	13	Parity error from CPU1 detected among signals protected by ParitySingle. Detected only when Schizo is in Excalibur mode
CPU1_Parity_BiDi	12	Parity error from CPU1 detected among signals protected by ParityBidi. Detected only when Schizo is in Excalibur mode
CPU0_Parity_Single	11	Parity error from CPU0 detected among signals protected by ParitySingle. Detected only when Schizo is in Excalibur mode
CPU0_Parity_BiDi	10	Parity error from CPU0 detected among signals protected by ParityBidi. Detected only when Schizo is in Excalibur mode
Safari_CIQ_Timeout	9	Transaction at head of CIQ longer than timeout interval
Safari_LPQ_Timeout	8	Transaction at head of LPQ longer than timeout interval
Safari_SFPQ_Timeout	7	Transaction at head of SFPQ longer than timeout interval
Safari_UFPQ_Timeout	6	Transaction at head of UFPQ longer than timeout interval
Safari_Addr_Par_Err	5	Safari Address Parity Error
Safari_Unmapped_Err	4	MappedIn snoop input not seen for transaction initiated by Schizo
Reserved	3	Reserved, read-only bits, read as 0.
Safari_BusError_DStat	2	Schizo received a Safari data packet for a non-cacheable DMA read with DStat indicating a Bus Error
Safari_Timeout_DStat	1	Schizo received a Safari data packet for a non-cacheable DMA read with DStat indicating a Timeout
Safari_Illegal_DStat	0	Schizo received a Safari data packet with an illegal DStat value. Illegal DStat combinations consist of BusError or Timeout DStat received in response to anything other than an RIO or RBIO transaction. Note that Schizo does not check PIOW's to UPA for this error.

Note – When CPU1 is not present, CPU1_Parity_Single and CPU1_Parity_BiDi must not be enabled. Without the CPU generating correct parity false parity errors will be logged, and if enabled will be reported causing system problems.

4.2.5 ECC Control Register

The Safari data path is protected by an Error Correcting Code. Each 128 bits of data is protected by a 9-bit SEC-DED-S4ED ECC field. The Mtag field is also protected by a 4-bit SEC-DED ECC field.

The ECC Control Register controls Schizo's handling of the Safari ECC fields, and associated error interrupts. In addition to the enables that are contained here, the Safari bus has a mechanism for dynamically signalling whether ECC is to be checked (via the DStat field). ECC must be enabled both in the ECC Control Register and the DStat field in order for Schizo to perform ECC checking.

Table 4-12 ECC Control Register

Bits	Field	Description	Reset Value	Type
63	ECC_EN	ECC Enable. ECC is checked on incoming data when set to one, and DStat allows it. ECC is always generated on outgoing data.	0	R/W
62	UE_INTEN	Uncorrectable error interrupt enable. When set to one a UE interrupt will be generated if an uncorrectable error is detected.	0	R/W
61	CE_INTEN	Correctable error interrupt enable. When set to one a CE interrupt will be generated if a correctable error is detected. The data is automatically corrected by the Schizo, so this interrupt is provided primarily for logging.	0	R/W
60:19	Rsvd	Reserved. Read as zero.	0x000 0000 0000	R
18	FMT	Force MTag ECC. For diagnostics only. When this bit is set to one the MTag ECC is copied from the FMECC field rather than being generated by Schizo's ECC logic.	0	R/W
17:14	FMECC	Forced MTag ECC. ECC value for all outgoing MTag when the FMT bit is set to one.	0x0	R/W
13	FMD	Force Data ECC. For diagnostics only. When this bit is set to one, the data ECC (for each 128 bits of data) is copied from the FDECC field instead of being generated by Schizo's ECC logic.	0	R/W
12:4	FDECC	Forced Data ECC. ECC value for all outgoing data when the FMD bit is set to one.	0x000	R/W
3:0	Rsvd	Reserved. Read as zero.	0x0	R

Note – The timing of changes to this register when a PIO write is performed is somewhat indeterminate. If software wants to ensure that a change takes effect before proceeding, it should follow the PIO write by a PIO read of this register.

The following table shows how the ECC_EN and UE_INTEN/CE_INTEN controls the ECC checking, error handling in Schizo. Further details on how transactions with errors are handled can be found in Chapter 5, “Error Handling.”

Table 4-13 ECC Error Reporting

ECC_EN	Safari DStat	INTEN	Description
0	X	X	No ECC checking and reporting, every Safari transaction proceed as if there is no ECC error.
1	1-3	X	Incoming Safari data is either invalid or does not have valid ECC, so ECC is not checked.
1	0	0	ECC checking is done. If an error is detected, error is logged in AFSR/AFAR but no interrupt is generated. Software should clear error status before enabling interrupt.
1	0	1	ECC checking is done. If an error is detected, error is logged in AFSR/AFAR and Schizo generates an interrupt.

4.2.6 *Correctable and Uncorrectable Error Asynchronous Fault Status Registers*

These registers (CE AFSR and UE AFSR) log correctable and uncorrectable ECC errors detected by the Safari interface in Schizo. ECC errors may occur on PIO writes, DVMA reads and partial DVMA writes which incur a read-modify-write operation.

“Primary” errors correspond to the first occurrence of a correctable ECC error logged into the CE AFSR and to the first occurrence of an uncorrectable ECC error logged into the UE AFSR. “Secondary” errors corresponds to errors which occur when a primary error is already logged and has not been cleared. This implies that a secondary error bit is set if one of the primary error bit is already set.

The order which defines primary and secondary error is the Safari data bus order which is not guaranteed to be the same order as the request. It should also be noted that ECC is checked only for data which is brought on chip, not necessarily all data transferred on the Safari data bus.

Secondary errors are cumulative, this means that more than one secondary error bit can be set if two or more errors occurred on different ECC words while a primary error was already logged. It also means that in the case where only one secondary error bit is set, more than one uncorrectable error may have occurred. It should be noted that only a single primary error bit can be set.

UPA64S accesses are treated differently as by definition UPA64S does not support ECC. PIO writes to UPA64S devices are received by Schizo with ECC. The ECC is checked by Schizo before being stripped off when the transaction is forwarded onto the UPA64S bus. Error logging, however, is incomplete when an ECC error is detected for a PIO write targeting a UPA64S device. The address is not recorded and some fields identifying the Safari transaction are not recorded either (see tables below).

The Correctable and Uncorrectable Error Asynchronous Fault Status registers have the following format:

Table 4-14 UE/CE Asynchronous Fault Status Registers

Bits	Field	Description	Reset Value	Type
63	P_PIO	Primary Error (CE or UE) on PIO. Set to one when the error is detected.	X	R/W1C
62	P_DRD	Primary Error (CE or UE) on DVMA Read. Set to one when the error is detected.	X	R/W1C
61	P_DWR	Primary Error (CE or UE) on partial DVMA write which incurred a load of the merge buffer. Set to one when the error is detected.	X	R/W1C
60	S_PIO	Secondary Error (CE or UE) on PIO. Set to one when the error is detected.	X	R/W1C
59	S_DMA	Secondary Error (CE or UE) on DVMA Read or partial Write which incurred a load of the merge buffer. Set to one when the error is detected.	X	R/W1C
58	Rsvd	Reserved. Read as 0.	0	R
57:56	ErrPNDG	Error Pending. These bits are zero when the error log is stable. When the error log is in the process of being updated this field is non-zero. Any PIO read which returns a non-zero value should be retried until this field is clear. (ErrPDNG[57] is related to P_PIO and ErrPNDG[56] is related to P_DRD and P_DWR.)	0x0	R
55:42	Rsvd	Reserved. Read as 0.	0x0000	R
41:32	MASK	10 bit Safari Mask (includes ByteMask and DWordMask) for the Safari transaction on which the primary error was detected. Valid only if P_PIO is set and PARTIAL is set, and undefined for UPA64S PIO Write.	0xXXX	R
31:30	QW_OFFSET	Quad Word Offset inside the 64 byte block on which the primary error was detected. Undefined for UPA64S PIO Write.	0xX	R
29	Rsvd	Reserved. Read as 0.	0	R
28:24	AGENT_ID	Safari Agent ID of the device that initiated the transaction that had faulty data. Undefined for UPA64S PIO Write.	0xXX	R
23	PARTIAL	Partial Transaction. Set to one if the primary error was detected on a PIO write or DMA Read of I/O space which was for a byte, half-word, word or double-word in size. Undefined for UPA64S PIO Write.	X	R
22	OWNED_IN	Set to one if the primary error occurred on a coherent DVMA read or partial DVMA write and the data was OwnedIn was asserted for the transaction. Valid only for cacheable DMA reads.	X	R
21:20	Rsvd	Reserved. Read as zero.	0xX	R

Table 4-14 UE/CE Asynchronous Fault Status Registers

Bits	Field	Description	Reset Value	Type
19:16	MTagECCSynd	Syndrome for the failing MTag	0xX	R
15:13	MTAG	MTag value, (as received.)	0xX	R
12:9	Rsvd	Reserved. Read as zero.	0xX	R
8:0	ECCSynd	ECC Syndrome for the failing data.	0xXXX	R

4.2.7 *Correctable and Uncorrectable Error Asynchronous Fault Address Register*

These registers (CE AFAR and UE AFAR) log the physical address of the data on which a primary correctable or uncorrectable error occurred.

Table 4-15 UE/CE Asynchronous Fault Address Register

Bits	Field	Description	Reset Value	Type
63:44	Rsvd	Reserved. Read as zero	0	R
43	I/O_Mem_Cmd	I/O or Memory Safari Command. When this bit is set to one, the error was detected on a Safari I/O transaction. When this bit is set to zero, the error was detected on a Safari memory transaction.	X	R
42:4	Address	PA[42:4] or pre-decoded address on PIO writes (see Table 4-16)	0xXXXXXXXXXX	R
3:0	Rsvd	Reserved. Read as zero	0	R

The format of the address logged is dependent of the transaction on which the error occurred. For PIO writes a pre-decoded address is recorded. The format in this case is the following:

Table 4-16 Address Format for PIO Writes

UEAFAR[42:39]	UEAFAR[38:36]	UEAFAR[35:4]	Region
0xF	Reserved. Read as 0.	Undefined.	UPA64S Space
0x8	Reserved. Read as 0.	PA[35:4]	
0xA	Reserved. Read as 0.	PA[35:4]	
0xB	Reserved. Read as 0.	PA[35:4]	
0x4	Reserved. Read as 0.	PA[35:4]	PCI A Interface registers
0x6	Reserved. Read as 0.	PA[35:4]	PCI A Memory Address Space
0x7	Reserved. Read as 0.	PA[35:4]	PCI A Configuration and I/O Spaces
0x0	Reserved. Read as 0.	PA[35:4]	PCI B Interface registers
0x2	Reserved. Read as 0.	PA[35:4]	PCI B Memory Address Space
0x3	Reserved. Read as 0.	PA[35:4]	PCI B Configuration and I/O Spaces
0xC	Reserved. Read as 0.	PA[35:4]	Safari and UPA64S interface registers

4.2.8 Safari Energy Star Control Register

Table 4-17 Safari Energy Star Control Register

Bits	Field	Description	Reset Value	Type
63:6	Rsvd	Reserved. Read as zero.	0	R
5	1/32 Speed	Operate Safari bus and interface logic at 1/32nd input frequency.	0	R/W
4:2	Rsvd	Reserved. Read as zero.	0	R
1	1/2 Speed	Operate Safari bus and interface logic at 1/2 the frequency of the input clock.	0	R/W
0	Full Speed	Operate Safari bus and interface logic at the same frequency as the input clock.	1	R/W

One and only one bit shall be set at a time.

Note – This register is writable only when the PLL is operating in normal mode. When S_PLL_BYPASS is asserted this register is held in its reset state.

4.2.9 Safari Soft Pause Register

Table 4-18 Safari Soft Pause Register

Bits	Field	Description	Reset Value	Type
63:1	Rsvd	Reserved. Read as zero.	0	R
0	Soft Pause	Assert internal value of S_PAUSEIN_L.	0	R/W

Used during Dynamic Reconfiguration to inhibit DMA traffic from Schizo. Setting this bit causes the internal value of S_PAUSEIN_L to be asserted. Schizo will behave as if external flow control logic had asserted the signal, primarily it will drive idle commands when it receives the Safari bus grant. Other operations will continue as normal, but with latency increased by the pause period while waiting for Safari commands to be issued.

This register must be used only with the greatest care to avoid deadlocks and other system failures. Specific, but not complete, hints include reading the register value back after setting the bit to know the write has completed. And all other PIO accesses to Schizo, (preferably to the whole I/O subsystem,) should be inhibited between setting and clearing this bit to avoid deadlocks.

4.2.10 Queue Control Register

This register is provided for diagnostic purposes only (mainly for simulation).

Table 4-19 Queue Control Register

Bits	Field	Description	Reset Value	Type
63:34	Rsvd	Reserved. Read as zero.	0	R
33	PCIA IS	Upbound queue. Forced empty is set to one.	0	R/W
32	Rsvd	Reserved. Read as zero.	0	R
31	PCIB IS	Upbound queue. Forced empty is set to one.	0	R/W
30	Rsvd	Reserved. Read as zero.	0	R
29	Rsvd	Reserved.	0	R/W
28	Rsvd	Reserved. Read as zero.	0	R
27	MB IS	Upbound queue. Forced empty is set to one.	0	R/W
26	MB RS	Downbound queue. Forced empty is set to one.	0	R/W
25	CSR IS	Upbound queue. Forced empty is set to one.	0	R/W
24	CSR RS	Downbound queue. Forced empty is set to one.	0	R/W
23:18	Rsvd	Reserved. Read as zero.	0	R
17	SWIDQ	Forced empty is set to one	0	R/W
16	UWIDQ	Forced empty is set to one	0	R/W
15:14	Rsvd	Reserved. Read as zero.	0	R
13	Rsvd	Reserved. Read as zero.	0	R
12	DDIQ	Forced empty is set to one	0	R/W
11	Rsvd	Reserved. Read as zero.	0	R
10	PDIQ	Forced empty is set to one	0	R/W
9	UDOQ	Forced empty is set to one	0	R/W
8	UDIQ	Forced empty is set to one	0	R/W
7:5	Rsvd	Reserved. Read as zero.	0	R
4	SRQ	Forced empty is set to one	0	R/W
3	LPQ	Forced empty is set to one	0	R/W
2	UPFQ	Forced empty is set to one	0	R/W
1	SFPQ	Forced empty is set to one	0	R/W
0	CIQ	Forced empty is set to one	0	R/W

The write timing of these bits is imprecise. Any write should be followed by a read to check for completion.

WIOs to clear the SFPQ, CIQ and UFPQ bits are not guaranteed to execute. When the SFPQ bit is set WIO transactions will not be processed. When the CIQ or UFPQ bits are set and the queues fill, the PAUSE flow control mechanism will be asserted and no transactions can be sent on the Safari bus.

4.2.11 Safari DTag Diagnostic Registers

The DTags are the dual tags that are used to maintain consistency on the contents of the merge buffer. The merge buffer is a small I/O cache of eight 64 bytes entries. It is used for any partial DMA writes (quantities of less than 64 bytes) to cacheable addresses. When an entry is allocated inside the merge buffer, Schizo assumes ownership of that line, and uses the DTags to snoop requests on the Safari Address and Command bus.

The DTags are made accessible for diagnostic purposes. It should be noted that read latency is large when compared to the snoop delay. If Safari transactions are changing the values, the value read may differ significantly from its value when the RIO was on the bus.

Table 4-20 DTag Diagnostic Register

Bits	Fields	Description	Reset Value	Type
63	V	Valid when set to one.	0	R
62:60	Rsvd	Reserved. Read as zero.	0	R
59:56	Id	DMA Id	0x0	R
55:43	Rsvd	Reserved. Read as zero.	0	R
42:6	PA[42:6]	Physical Address	0x00 0000 0000	R
5:0	Rsvd	Reserved. Read as zero.	0	R

The DMA Id identifies the Partial DMA Write using the merge buffer. (It equals AtransID[3:0] used the RTO and WB commands.)

Note – There are 8 separate DTag diagnostic registers (one for each merge buffer entry) aligned on 16 byte boundaries although they are only 8 byte long.

4.2.12 Safari Debug Registers

Schizo includes some limited support for debug. These registers control the Safari and UPA specific logic and expose select internal nodes for added visibility. The debug logic and these registers are not guaranteed from version to version. Practically they need to be used with RTL code in hand and never should be enabled in normal operation (as some may cause strange system specific problems.)

These registers are all part of the JTAG Shadow Scan chain, and typically more usefully accessed there.

The first controls the Safari Debug mux selecting 8 internal nodes to be driven on the, normally input, Z_AID[4:0] and Z_SYSCODE[2:0] pins.

Table 4-21 Safari Debug Mux Select Register (0x414000)

Bits	Fields	Description	Reset Value	Type
63:34	Rsvd	Reserved. Read as zero.	0	R
33	UPA_En	Enable UPA Debug mode	0	R/W
32	Saf_En	Enable Safari Debug Mux outputs	0	R/W
31:27	Rsvd	Reserved. Read as zero.	0	R
26:24	Mod_Sel0	Select for Safari Module driving Mux0	0	R/W
23:21	Rsvd	Reserved. Read as zero.	0	R
20:16	Sel0	Select for Mux0	0	R/W
15:11	Rsvd	Reserved. Read as zero.	0	R
10:8	Mod_Sel1	Select for Safari Module driving Mux1	0	R/W
7:5	Rsvd	Reserved. Read as zero.	0	R
4:0	Sel1	Select for Mux1	0	R/W

Many of the internal state bits are made visible. These bits can be read with JTAG or at these PIO addresses. Since these are not cycle accurate and can only be read with relatively high latency they must be taken as approximations.

Table 4-22 Safari Debug Register (0x414008)

Bits	Fields	Description	Reset Value	Type
63:19	Rsvd	Reserved. Read as zero.	0	R
18:11	DTag Valid	Which DTag entries are in use.	-	R
10	UFPQ_HW	UFPQ reached High Water mark	0	R
9	CIQ_HW	CIQ reached High Water mark	0	R
8	UWIDQ_HW	UWIDQ reached High Water mark	0	R
7	PDOQ_Full	PDOQ is full	0	R
6	Rsvd	Reserved. Read as zero.	0	R
5	B_RS_Full	MIT RS FIFO to PCI-B is Full	0	R
4	A_RS_Full	MIT RS FIFO to PCI-A is Full	0	R
3	MB_IS_Full	MIT IS FIFO to Merge Buffer is Full	0	R

Table 4-22 Safari Debug Register (0x414008)

Bits	Fields	Description	Reset Value	Type
2	MB_RS_Full	MIT RS FIFO to Merge Buffer is Full	0	R
1	R_IS_Full	MIT IS FIFO to Safari Registers is Full	0	R
0	R_RS_Full	MIT RS FIFO to Safari Registers is Full	0	R

The bits in the following registers are expected to be zero when the Safari interface is idle. (Note that PIO Reads will make the Safari interface not idle.)

Table 4-23 Safari Idle0 Register (0x414010)

Bits	Fields	Description	Reset Value	Type
63:22	Rsvd	Reserved. Read as zero.	0	R
21	DTags Valid	One or more DTags is in use.	-	R
20	CTags Valid	One of more CTags, Merge Buffer lines, is in use.	0	R
19	ORQ Empty	ORQ is Empty when 0	0	R
18	SRQ Empty	SRQ is Empty when 0	0	R
17	CIQ Empty	CIQ is Empty when 0	0	R
16	LPQ Empty	LPQ is Empty when 0	0	R
15	SFPQ Empty	SFPQ is Empty when 0	0	R
14	UWIDQ Empty	UWIDQ is Empty when 0	0	R
13	SWIDQ Empty	SWIDQ is Empty when 0	0	R
12	SRQ Empty	SRQ is Empty when 0	0	R
11:10	Rsvd	Reserved. Read as zero.	0	R
9:8	B_PIO_Count	Pending PCI-B PIO Count	0	R
7:6	A_PIO_Count	Pending PCI-A PIO Count	0	R
5:2	Rsvd	Reserved. Read as zero.	0	R
1	MB_IS_Empty	MIT IS FIFO from Merge Buffer is Empty when 0	0	R
0	R_IS_Empty	MIT IS FIFO from Safari Registers is Empty when 0	0	R

Table 4-24 Safari Idle1 Register (0x414018)

Bits	Fields	Description	Reset Value	Type
63	B_IS_Empty	MIT IS FIFO from PCI-B is Empty when 0	0	R
62	A_IS_Empty	MIT IS FIFO from PCI-A is Empty when 0	0	R
61	Rsvd	Reserved. Read as zero.	0	R
60	FSM_MB_Idle	DMA FSM's MB Interface is Idle when 0	0	R
59	FSM_R_Idle	DMA FSM's Safari Registers Interface is Idle when 0	0	R
58	FSM_B_Idle	DMA FSM's PCI-B Interface is Idle when 0	0	R
57	FSM_A_Idle	DMA FSM's PCI-B Interface is Idle when 0	0	R
56	Rsvd	Reserved. Read as zero.	0	R
55	MB_RS_Empty	MIT RS FIFO to MB is Empty when 0	0	R
54	R_RS_Empty	MIT RS FIFO to Safari Registers is Empty when 0	0	R
53	UDOQ_Empty	UDOQ FIFO is Empty when 0	0	R
52	UFPQ_Empty	UFPQ FIFO is Empty when 0	0	R
51	UPA_FSM_Idle	UPA FSM is Idle when 0	0	R
50	Pndg_P_Reply	UPA P_Reply Pending	0	R
49	UDIQ_Pkt_Pndg	UPA Write Data Pending	0	R
48:41	PDIQ Valid	PDIQ Buffer valid flags	0	R
40:25	UDIQ Valid	UDIQ Buffer valid flags	0	R
24	PDOQ Empty	PDOQ FIFO is Empty when 0	0	R
23:16	DDOQ Valid	DDOQ Buffer valid flags, statically allocated 23:20 for PCI-B and 19:16 for PCI-A	0	R
15:12	Rsvd	Reserved. Read as zero.	0	R
11:4	DDIQ Valid	DDIQ Buffer valid flags, statically allocated 11:8 for PCI-B and 7:4 for PCI-A	0	R
3:0	Rsvd	Reserved. Read as zero.	0	R

Note that DDOQ and DDIQ valid bits will not be cleared on some errors and can therefore appear busy even after all transaction have ended.

4.2.13 Safari Performance Control Register

Schizo includes two 32-bit counter that can be used to gather statistics on Safari related hardware events.

The type of events that are counted by the performance counters is specified through the Safari Performance Control register. The format is:

Table 4-25 Safari Performance Control Register

Bits	Fields	Description	Reset Value	Type
63:16	Rsvd	Reserved. Read as zero.	0	R
15:11	Cnt1 Select	Event selection for counter 1	0	R/W
10:9	Rsvd	Reserved. Read as zero.	0	R
8:4	Cnt0 Select	Event selection for counter 0	0	R/W
3:0	Rsvd	Reserved. Read as zero.	0	R

The list of events with their selection codes is:

Table 4-26 Safari Events Selection codes:

Code	Safari Event
0x00	Counting Disabled.
0x01	Safari Bus cycles
0x02	Cycles Pause is asserted by this Schizo (flow control).
0x03	Foreign Coherent transactions (i.e. RTO, RTOR, RTS, RTSR, RTSM, RS, RSR, WS with an ID other than this Schizo.)
0x04	Foreign coherent hits (i.e active sharing with the contents of the merge buffer).
0x05	Schizo's own coherent transactions (RTO, RTOR, RS, RSR, & WS with my ID)
0x06	Schizo's coherent hits in its own Merge Buffer (i.e transfers requiring self-copybacks).
0x07	Foreign I/O transactions (i.e. RIO, RBIO, WIO, & WBIO with an ID other than mine.)
0x08	Foreign I/O hits (i.e. transaction in I/O space mapped to this Schizo).
0x09	Partial writes in merge buffer (RTO, RTOR with my ID.)
0x0a	Interrupts issued. (INT with my ID.)
0x0b	(Reserved)
0x0c	PIO accesses to Schizo's internal registers
0x0d	UPA accesses
0x0e	PCI A accesses
0x0f	PCI B accesses
0x10	(Reserved)

Table 4-26 Safari Events Selection codes:

Code	Safari Event
0x11	Cycles of Pause in System (i.e. cycles we receive S_PAUSEIN_L asserted.)
0x12	DVMA Read. (RS, RSR, RIO, RBIO with my ID.)
0x13	DVMA Write. (RTO, RTOR, WS, WIO, WBIO with my ID.)
0x14	ORQ Full.
0x15	ZData Input. (Cycles I receive data on Z_DATA_L pins.)
0x16	ZData Output. (Cycles I drive data on Z_DATA_L pins.)
0x17 - 0x1F	Reserved

Note – Some transactions will be counted twice. These include Interrupts which are NAck'd and reissued, and in SSM systems all commands which are reissued (by asserting Safari Owned line. R_* commands are not counted.)

4.2.14 Safari Performance Counters Register

The performance counters are wrap around counters. They are 32 bit wide so with a 150 Mhz Safari clock they wrap in about 28.6 seconds when counting bus cycles. Note that since the only legal access size is 8 bytes, both count fields must be updated (e.g. set to 0) together. Also note that the values are not reset automatically when the counted events are changed.

Table 4-27 Safari Performance Counters Register

Bits	Fields	Description	Reset Value	Type
63:32	Cnt1	Counter 1	0	R/W
31:0	Cnt10	Counter 0	0	R/W

4.3 UPA Leaf

Table 4-28 UPA Register Offsets

Register	Offset	Access Size
UPA Slot0 Configuration Register	0x48.0000	8 bytes
UPA Slot1 Configuration Register	0x48.0008	8 bytes
UPA Interface Configuration Register	0x48.0010	8 bytes
UPA Energy Star Control Register	0x48.0018	8 bytes

4.3.1 UPA64S Slot0 and Slot 1 Configuration Registers

The UPA64S Slot Configuration registers are provided for compliance with the UPA specification. The reset bit has moved to the UPA64S Interface Configuration register.

The format for both Slot Configuration registers is:

Table 4-29 UPA64S Slot 0 and Slot 1 Configuration Register

Bits	Fields	Description	Reset Value	Type
63	Slot Empty	This field is used to detect the presence of a board in the slot. It is set to one if the slot is empty and cleared to zero if a board is detected. If a slot is empty its corresponding Address Match and Mask registers are considered invalid Also JTAG Shadow R/W.	-	R
62:32	Rsvd	Reserved. Read as zero.	0	R
31	Rsvd	Reserved. Read as one.	1	R
30:28	Rsvd	Reserved. Read as zero.	0	R
27:24	SPRQS	Slave P_Req queue size. Schizo maintains a single count and uses the lower SPRQS value of slots with present devices. Schizo also implements a maximum of 7 outstanding transactions and uses 7 when SPRQS is set to a greater value. SPRQS may not be decreased.	0x1	R/W
23:18	SPDQS	Slave Port Data queue size. Schizo does not use this parameter. It always assumes that SPDQS = 4 * SPRDQS.	0x4	R
17:16	Rsvd	Reserved. Read as zero.	0	R
15	SQUEN	Slave Queues Enable. This bit must be set to one when writing this register. It is always read as zero.	0 (R0)	W
14	OneRead	Read as one.	1	R
13:0	Rsvd	Reserved. Read as zero.	0	R

I/O space address decoding (see Safari Address Match and Mask Registers) is qualified by the presence of a device in the UPA slot. That is, when Slot Empty is set the corresponding address decoder is disabled and MappedOut is not asserted for the address region. This prevents transactions from timing out because no device is present to respond.

4.3.2 *UPA64S Interface Configuration Register*

The format for the UPA64S Interface Configuration register (UPA64S_ICR) is:

Table 4-30 UPA64S Interface Configuration Register

Bits	Fields	Description	Reset Value	Type
63:2	Rsvd	Reserved. Read as zero.	0	R
1	UPA_POK	Power OK. The UPA_POWER_OK output of Schizo always reflects the value of this bit.	0	R/W
0	UPA_RST_L	Reset. The UPA_RESET_L output of Schizo always reflects the exact value of this bit (no inversion).	0	R/W

4.3.2.0.1 *Reset*

Whenever Schizo is reset (by any reset condition), these two bits are cleared. Any UPA64S devices are held in reset until these bits are set to 1 by software. To ensure that UPA64S devices come out of reset correctly with their PLLs locked, the following sequence is required:

- store @UPA64S_ICR, 0x2
- wait for PLL to lock (typically ~1 ms is sufficient)
- store @UPA64S_ICR, 0x3

4.3.2.0.2 *Energy Star*

The reset bits in the UPA64S_ICR can also be used for (heavy-hammer) Energy Star control. In Energy Star mode, any UPA64S framebuffers probably don't need to be active, and can be held in a reset and powered down state. In order to enter this state, the following sequence should be used:

- store @UPA64S_ICR, 0x2
- load @UPA64S_ICR (used to provide guaranteed short delay, load must be blocking)
- store @UPA64S_ICR, 0x0

When coming out of Energy Star mode, the sequence for coming out of reset above should be used.

I/O space address decoding (see Safari Address Match and Mask Registers) is qualified by the assertion of UPA reset. That is, when UPA_RST_L is clear the corresponding address decoders are disabled and MappedOut is not asserted for either UPA address region. This prevents transactions from timing out because no device is functioning and able to respond.

4.3.3 *UPA Energy Star Control Register*

Is used to affect the operating frequency of the on-chip UPA logic when the external UPA bus is forced idle with UPA_RST_L. This allows a power reduction.

Table 4-31 UPA Energy Star Control Register

Bits	Field	Description	Reset Value	Type
63:7	Rsvd	Reserved. Read as zero.	0	R
6	1/64 Speed	Operate UPA interface logic at 1/64 the frequency of the input clock.	0	R/W
5:2	Rsvd	Reserved. Read as zero.	0	R
1	1/2 Speed	Operate UPA interface logic at 1/2 the frequency of the input clock.	0	R/W
0	Full Speed	Operate UPA interface logic at the same frequency as the input clock.	1	R/W

One and only one bit shall be set at a time.

4.4 PCI Leaf

Note – Schizo contains two nearly identical copies of a PCI Leaf Block. The programmer's model for both copies is identical. Register addresses are shown with respect to the base address PCI_CSRBase, whose real value should be one of the two base addresses PCI-A_CSRBase or PCI-B_CSRBase. Registers which are in PCI Configuration Space are shown with respect to PCI_ConfigBase, which again should be the appropriate PCI-A_ConfigBase or PCI-B_ConfigBase address.

4.4.1 PCI Bus Module

Within a PCI Leaf, the block directly responsible for the PCI transactions and protocol is the PCI Bus Module (PBM), which has a set of control/status registers. Some of these registers control aspects of Schizo's PCI operations that are not completely defined, or left implementation dependent by the PCI specification. These PBM registers are placed into the PCI_CSRBase address region. The PBM also has a number of registers in PCI Configuration Space which are defined by the PCI specification.

Table 4-32 Offset of PBM Registers

Register	Address	Access Size
PCI Control/Status Register	PCI_CSRBase+0x0.0000.2000	8 bytes
PCI AFSR	PCI_CSRBase+0x0.0000.2010	8 bytes
PCI AFAR	PCI_CSRBase+0x0.0000.2018	8 bytes
PCI Diagnostic Register	PCI_CSRBase+0x0.0000.2020	8 bytes
PCI Energy Star Register	PCI_CSRBase+0x0.0000.2028	8 bytes

4.4.1.1 PCI Control/Status Register

.

Table 4-33 PCI Control and Status Register

Field	Bits	Reset	Description	R/W
BUS_UNUSABLE	63	0	PCI bus has been left in an inconsistent state, and is expected to be unusable. While this bit is set, Schizo will not perform any PIO operations to the PCI bus, but will instead treat each PIO as an error. Should only be cleared when PCI bus is known to be usable again.	R/W1C
Reserved	62:52	0	Reserved, read as 0	R
DMA_WR_PERR	51	0	Set to 1 if a DMA write parity error is detected.	R/W1C

Table 4-33 PCI Control and Status Register

Field	Bits	Reset	Description	R/W
ERR_SLOT	50:48	0	Error slot. Tracks the current master on the PCI bus. Locks whenever a PCI interrupt is signalled. Set to the encoded slot number of the current PCI master device: 0x0 = Device receiving GNT[0] ... 0x5 = Device receiving GNT[5] 0x6 = Schizo 0x7 = Reserved	R
Reserved	47:39	0	Reserved, read as 0	R
PCI_TTO_ERR	38	0	PCI TRDY# Timeout error Set to 1 when a TRDY# timeout is detected during any PIO by Schizo	R/W1C
PCI_RTRY_ERR	37	0	PCI Excessive Retry error Set to 1 if the maximum retries are exceeded during any PIO.	R/W1C
PCI_MMU_ERR	36	0	PCI IOMMU error Set to 1 if an IOMMU error is detected for any DMA.	R/W1C
PCI_SBH_ERR	35	0	PCI streaming byte hole error Set to 1 whenever a byte hole is detected during a streaming DMA write.	R/W1C
PCI_SERR	34	0	Set when SERR# signal is sampled asserted on the PCI bus	R/W1C
PCI_SPEED	33	-	PCI bus speed. 0 = Schizo clock / 2 1 = Schizo clock The value of this bit reflects the status of the bus speed input pin. It is calculated by motherboard circuitry at power-on, based on capabilities of plugged in devices	R
Reserved	32:26	0	Reserved, Read as 0.	R
PTO	25:24	0	PCI Timeout interval. See Table 4-34 for definition.	R/W
Reserved	23:20	0	Reserved, Read as 0.	R
MMU_INT_EN	19	0	IOMMU error interrupt enable 0 = MMU errors will not cause a PCI error to be issued 1 = MMU errors will cause a PCI error to be issued	R/W

Table 4-33 PCI Control and Status Register

Field	Bits	Reset	Description	R/W
SBH_INT_EN	18	0	Streaming byte hole interrupt enable 0 = PCI error interrupt will not be issued for streaming byte hole errors 1 = PCI error interrupt will be issued for streaming byte hole errors, if ERRINT_EN is also set to 1.	R/W
ERRINT_EN	17	0	Enable PCI error interrupt. 0 = PCI error interrupt disabled 1 = PCI error interrupt enabled	R/W
ARB_PARK	16	0	PCI bus arbitration parking enable. 0 = no parking 1 = previous bus owner parked (including CPU)	R/W
Reserved	15:9	0	Reserved, read as 0.	R
PCI_RST	8	0	PCI bus reset. When set to 1, Schizo asserts the PCI_RST_L signal for this bus.	R/W
Reserved	7:6	0	Reserved, read as 0.	R
ARB_EN<5:0>	5:0	0	PCI DMA arbitration enable. One independent bit for each potentially supported master device on the bus. 0 = Bus requests from corresponding PCI device are ignored 1 = Bus requests from corresponding PCI device are honored.	R/W

4.4.1.1.1 PCI Timeouts

Schizo is capable of detecting several different types of timeouts as error conditions on the PCI bus:

- Discard Timeout (DTO) - If a PCI master has not reissued a PCI Delayed Read Request within a certain number of cycles, Schizo discards any data it may have for the read transaction, and signals this error.
- TRDY# Timeout (TTO) - When Schizo is a PCI master and a slave that has already claimed the cycle with DEVSEL# takes too many cycles to assert TRDY#, Schizo will attempt to terminate the transaction, will signal this error, and will mark the PCI bus as unusable via the BUS_UNUSABLE bit.
- Max retries exceeded (RTRY) - When Schizo is a PCI master doing a PIO read or write, and the target device terminates the transaction with a retry more than the specified number of times in a row, Schizo will abort the transaction and signal this error.

The durations of each of these timeouts are controlled globally by the PTO field in the PCI Control/Status Register, which has the following definition:

Table 4-34 PCI Timeout Intervals

PTO<1:0>	Maximum Retries (in # retries)	Interval Timer (in # bus cycles)	Intended Purpose
00	∞ (disabled)	∞ (disabled)	Reset value. Used for debug/boot.
01	2^14	2^15	Standard value.
10	2^11	2^12	Bringup/lab testing
11	2^7	2^9	Simulation

The Interval Timer is used for both the DTO and TTO timeouts in an inexact fashion. The Interval Timer is always running. Under the proper conditions, if the Interval Timer reaches its maximum value twice, then Schizo will signal the appropriate error. As a result, each of these timeouts can have the following range of values:

(Max Interval Timer) < DTO/TTO timeout <= 2 * (Max Interval Timer)

Note – The global disable provided by PTO=0x0 is in addition to individual disables that may exist (e.g. for max retries and TTO error).

4.4.1.2 *PCI Asynchronous Fault Status/Address Registers*

PCI AFSR/AFAR record error information related to PIO writes to PCI slave devices. Only asynchronous errors reported through interrupt are recorded in these registers. Asynchronous errors include any PIO write access terminated by Master Abort, Target Abort, or excessive retries, as well as any PIO write during which a data parity error was signaled on the PCI bus. Although status bits for Master Abort, Target Abort and Parity Error exist in the PCI Configuration Registers for each PBM, they are duplicated here to provide the additional functionality of identifying which error occurred first in the case of multiple errors, and associating an address with that error.

Two sets of status bits are defined in this register. Bits <63:60> are the primary error status and bits <59:56> are the secondary error status. One and only one of the primary error status can be set at any time. Primary error status can be set only when

- none of the primary error condition exists prior to this error OR
- new error detected at the same time software is clearing the primary error.

Secondary bits are set whenever a primary bit is set (one and only one primary bit can be set at a time). The secondary bits are cumulative and always indicate that information has been lost as no address information has been captured. Setting of the primary error bits is independent.

The AFAR and bits <47:37> of AFSR logs address and status of the primary PCI PIO error. Further PCI PIO error will not be logged into these bits until software clears the primary error, which makes the AFAR and part of the AFSR available to log new error. An interrupt is generated, if enabled, whenever the AFAR logs the new error address.

Table 4-35 PCI AFSR

Field	Bits	Reset	Description	R/W
P_MA	63	X	Set if primary error detected is Master Abort	R/W1C
P_TA	62	X	Set if primary error detected is Target Abort	R/W1C
P_RTRY	61	X	Set if primary error detected is excessive retries	R/W1C
P_PERR	60	X	Set if primary error detected is data parity error	R/W1C
P_TTO	59	X	Set if primary error detected is TRDY# timeout	R/W1C
P_UNUSABLE	58	X	Set if primary error detected is BUS_UNUSABLE error	R/W1C
S_MA	57	X	Set if secondary error detected is Master Abort	R/W1C
S_TA	56	X	Set if secondary error detected is Target Abort	R/W1C
S_RTRY	55	X	Set if secondary error detected is excessive retries	R/W1C
S_PERR	54	X	Set if secondary error detected is data parity error	R/W1C
S_TTO	53	X	Set if secondary error detected is TRDY# timeout	R/W1C
S_UNUSABLE	52	X	Set if secondary error detected is BUS_UNUSABLE error	R/W1C
Reserved	51:42	0	Reserved, read as 0	R
MASK	41:32	X	Safari Mask of failed primary transfer (MASK<9:8> is Safari DWordMask<1:0>, and MASK<7:0> is Safari ByteMask<7:0>). Only valid if BLK is 0	R
BLK	31	X	Set to 1 if failed primary transfer was a block read or write	R
ConfigSpace	30	X	Set to 1 if failed primary transaction was to PCI Configuration Space.	R
MemorySpace	29	X	Set to 1 if failed primary transaction was to PCI Memory Space	R
IOSpace	28	X	Set to 1 if failed primary transaction was to PCI IO Space	R
Reserved	27:0	0	Reserved, read as 0	R

Table 4-36 PCI AFAR

Field	Bits	Reset	Description	R/W
Reserved	63:32	0	Reserved, read as 0.	R
PA_Offset	31:0	X	Offset of physical address of error transaction.	R

4.4.1.3 PCI Diagnostic Register

Table 4-37 PCI Diagnostic Register

Field	Bits	Reset	Description	R/W
Reserved	63:11	0	Reserved, read as 0.	R
STOP_DATA	10	0	Stop data from being written to memory when a parity error is detected. When set = 0, parity errors that Schizo detects during non-streaming DMA writes will allow the data to pass to Safari. When set to 1, parity errors that Schizo detects during DMA writes will prevent data from being passed to Safari (non-streaming only). For streaming DMA's regardless of the value of STOP_DATA, data is passed to Safari. In either case, the DPE bit will get marked and an interrupt generated, but NO Target abort will be generated. ECC is never actually corrupted.	R/W
DIS_BYPASS	9	0	Disable MMU Bypass mode When set to 1, Schizo will not respond to PCI Dual-Address Cycles, thus disallowing MMU bypass mode.	R/W
DIS_TTO	8	0	Disable TRDY# timeout errors. When set to 1, Schizo will not terminate any transactions as master due to excessive delays in TRDY#.	R/W
DIS_RTY_ARB	7	0	Disable retry arbitration priority. When set to 1, Schizo will not give PCI devices that have been previously retried priority over other devices when arbitrating for the PCI bus.	R/W
DIS_RETRY	6	0	Disable retry limit. When set to 1, Schizo will not abort PIO operations after 16,384 retries, but will continue indefinitely.	R/W
DIS_INTSYNC	5	0	Disable DMA write / interrupt synchronization. When set to 1, interrupts will not wait until associated DMA is complete before proceeding.	R/W
Reserved	4	0	Reserved, read as 0.	R
I_DMA_D_PAR	3	0	Invert DMA data parity 0 = Correct parity asserted 1 = Incorrect parity asserted for all PCI DMA read data phases. Both the regular parity signal and the 64-bit parity extension are affected.	R/W

Table 4-37 PCI Diagnostic Register

Field	Bits	Reset	Description	R/W
I_PIO_D_PAR	2	0	Invert PIO data parity 0 = Correct parity asserted 1 = Incorrect parity asserted for all PCI PIO write data phases.	R/W
I_PIO_A_PAR	1	0	Invert PIO address parity 0 = Correct parity asserted 1 = Incorrect parity asserted for all PCI PIO address phases.	R/W
Reserved	0	0	Reserved, read as 0.	R

4.4.1.3.1 Loopback Mode

There is a diagnostic Loopback mode in which a PCI Leaf in Schizo can act as both the initiator and target of a PCI transaction. This is not, however, enabled by a bit in the PCI Diagnostic Register. It is instead enabled by simply allocating a 4 Gb address range for PCI Memory Space via the appropriate PCI-{A,B}_Mem Address Mask Register in the Safari Interface Block (since the upper 2 Gb of PCI Memory Space is reserved for DMA, 2Gb is normally the largest address space needed for mapping PIOs to PCI Memory Space). If the entire 4 Gb address space is mapped, an outgoing PIO in the upper 2 Gb of the address space will be sent to the PCI bus, but will also be interpreted by Schizo as a DMA transaction, and the PCI Leaf will respond accordingly.

Any transaction that can be both legally generated and accepted by Schizo may be looped back in this fashion (e.g. streamable, consistent, pass-through, non-cacheable).

4.4.1.4 PCI Energy Star (E) Register*

Table 4-38 PCI Energy Star Register

Field	Bits	Reset	Description	R/W
Reserved	63:1	0	Reserved, read as 0.	R
ESTAR_MODE	0	0	Energy Star Mode When this bit is set, Schizo will dynamically request the PCI clock frequency to change (by way of the A_SLOW_CLOCK and B_SLOW_CLOCK outputs) based on PCI bus activity.	R/W

4.4.1.4.1 Energy Star Mode

When the ESTAR_MODE bit is set in Schizo, Schizo dynamically changes the PCI clock frequency based on bus activity. If ESTAR_MODE is set, whenever the bus is idle (no current transactions, no pending arbitration requests), Schizo initiates a frequency slowdown by signalling the external clock generator by way of the A/B_SLOW_CLOCK outputs. At the same time, Schizo slows its internal clock down by a factor of 32.

While the clocks are in slow mode, no transactions are allowed on the bus. If a PCI arbitration request gets asserted, Schizo will deassert A/B_SLOW_CLOCK, bring its internal clock to full speed, wait until the PCI bus is stable again, and then issue a GNT# to the requesting device. This applies to the internal GNT# used for pio transactions as well.

4.4.1.5 *PBM Configuration Space*

The PBM also contains a configuration header whose format is specified by the PCI Specification. The registers in the configuration header are accessed in PCI Configuration Address Space. The PBM is considered to be device 0, function 0 on its PCI bus. After a reset, the PBM's Bus Number register is set to 0

Table 4-39 Default addresses PCI Leaf's internal PCI Configuration Registers

Register	Address
PBM Config Registers	PCI_ConfigBase+0x000000 - PCI_ConfigBase+0x0000FF

Note – These are the addresses in effect after reset. However, since the PCI bus number of the PBM can be changed by software, the actual offset for these spaces may be different than what is listed above.

Note – The PCI Configuration Address Space is a little-endian address space. When accessing configuration space registers, software should take advantage of one of the SPARC V9 little-endian support mechanisms to get proper byte ordering. These mechanisms include little-endian ASIs or MMU support for marking pages little-endian.

The table below lists the configuration header registers, as defined by the PCI specification and PCI System Design Guide. Several of the registers are not implemented in Schizo which is indicated by shading in the table. The rule used is that any optional register for which equivalent information exists elsewhere is not implemented..

Table 4-40 Configuration Space Header Summary

Register	Offset	Size
Required PCI device configuration header:		
Vendor ID	0x00	2 bytes
Device ID	0x02	2 bytes
Command	0x04	2 bytes
Status	0x06	2 bytes
Revision ID	0x08	1 byte
Programming I/F Code	0x09	1 byte
Sub-class Code	0x0A	1 byte

Table 4-40 Configuration Space Header Summary

Register	Offset	Size
Base Class Code	0x0B	1 byte
Cache Line Size	0x0C	1 byte
Latency Timer	0x0D	1 byte
Header Type	0x0E	1 byte
BIST	0x0F	1 byte
Base Address	0x10-0x27	Varies
Reserved	0x28-0x2F	n/a
Expansion ROM	0x30	4 bytes
Reserved	0x34-0x3B	n/a
Interrupt Line	0x3C	1 byte
Interrupt Pin	0x3D	1 byte
MIN_GNT	0x3E	1 byte
MAX_LAT	0x3F	1 byte
Optional bridge configuration header:		
Bus Number	0x40	1 byte
Subordinate Bus Number	0x41	1 byte
Reserved	0x42-0xFF	n/a
Disconnect Counter	Unspecified	1 byte
Bridge Command/Status	Unspecified	4 bytes
Bridge Memory Base Address	Unspecified	4 bytes
Bridge Memory Limit Address	Unspecified	4 bytes
DOS Read Attributes	Unspecified	2 bytes
DOS Write Attributes	Unspecified	2 bytes
Bridge I/O Base Address	Unspecified	2 bytes
Bridge I/O Limit Address	Unspecified	2 bytes

Note – The sizes listed in the table above are just the logical size for each register. Actual PIO access to the registers can be in any size from 1 to 4 bytes, provided the access doesn't span multiple 32-bit words.

4.4.1.5.1 *Vendor ID*

Read only, VendorID<15:0> = 0x108E.

4.4.1.5.2 *Device ID*

Read only, DeviceID<15:0> = 0x8001.

4.4.1.5.3 Command Register

Table 4-41 Command Register

Field	Bits	Reset	Description	R/W
Reserved	15:10	0	Reserved, read as 0.	R
FAST_EN	9	0	Enable fast back-to-back cycles to different targets. Hardwired to 0 (disabled).	R
SERR_EN	8	0	Enable driving of SERR# pin when 1.	R/W
WAIT	7	0	Enable use of address/data stepping Hardwired to 0 (disabled).	R
PER	6	0	Enable reporting of parity errors when 1.	R/W
VGA	5	0	Enable VGA palette snooping Hardwired to 0 (disabled).	R
MWI	4	0	Enables use of Memory Write & Invalidate Hardwired to 0 (disabled).	R
SPCL	3	0	Enables monitoring of special cycles Hardwired to 0 (disabled).	R
MSTR	2	1	Enables ability to be bus master Hardwired to 1 (enabled).	R
MEM	1	1	Enables response to PCI MEM cycles Hardwired to 1 (enabled).	R
IO	0	0	Enables response to PCI I/O cycles. Hardwired to 0 (disabled).	R

4.4.1.5.4 Status Register

Table 4-42 Status Register

Field	Bits	Reset	Description	R/W
DPE	15	X	Set if PBM detects a parity error	R/W1C
SSE	14	0	Set if PBM signalled a system error. This occurs if the PBM detects a PCI address parity error, or another device asserts SERR#.	R/W1C
RMA	13	0	Set if PBM receives a master-abort	R/W1C
RTA	12	0	Set if PBM receives a target-abort	R/W1C
STA	11	0	Set if PBM generates target-abort	R/W1C
DVSL	10:9	0x1	Timing of DEVSEL#. Hardwired to 01 (medium speed response)	R
DPAR	8	0	Set when parity error occurs while PBM is bus master, if PER in command register also set.	R/W1C

Table 4-42 Status Register

Field	Bits	Reset	Description	R/W
FASTCAP	7	1	Indicates ability to accept fast back-to-back cycles as target, when the back-to-back transactions are not to the same target. Hardwired to 1 (allowed)	R
UDF_SUPPORT	6	0	User Definable Feature Support Hardwired to 0 (no user definable features)	R
66MHZ_CAPABLE	5	-	Indicates ability to run at 66MHz clock speed. Hardwired to 1 (66MHz capable) for PBMA and 0 for PBMB.	R
Reserved	4:0	0	Reserved, read as 0	R

4.4.1.5.5 *Revision ID Register*

Read only, RevisionID<7:0> = 0x00. This register will always read as 0. The actual revision number for Schizo is contained in the Schizo Control/Status Register.

4.4.1.5.6 *Programming I/F Code Register*

Read only, ProgrammingIFCode<7:0> = 0x00.

4.4.1.5.7 *Sub-class Code Register*

Read only, SubclassCode<7:0> = 0x00. (Specifies host bridge device).

4.4.1.5.8 *Base Class Code Register*

Read only, BaseClassCode<7:0> = 0x06. (Specifies bridge device).

4.4.1.5.9 *Latency Timer Register*

This 8-bit read/write register specifies the value of the latency timer for the PBM as a bus master. Only the top five bits are implemented, giving a timer granularity of 8 PCI clocks. The bottom three bits will read as 0 and should be written as 0.

Table 4-43 Latency Timer Register

Field	Bits	Reset	Description	R/W
LAT_TMR_HI	7:3	0x00	Programmable portion of latency timer.	R/W
LAT_TMR_LO	2:0	0	Read only portion of latency timer. Hardwired to 0.	R

4.4.1.5.10 Header Type Register

Table 4-44 Header Type Register

Field	Bits	Reset	Description	R/W
MULTI_FUNC	7	0	Indicates whether the PBM is a multi-function PCI device. Hardwired to 0 (not multi-function).	R
HDR_TYPE	6:0	0	Defines layout of configuration header bytes 0x10-0x3F. Hardwired to 0 (the only defined value in PCI specification)	R

4.4.1.5.11 Bus Number

This 8-bit read/write register specifies the number of the PCI bus this bridge resides on. It's value upon reset is 0.

4.4.1.5.12 Subordinate Bus Number

This 8-bit read/write register specifies the highest subordinate bus number beneath this bridge. It's value upon reset is 0.

4.4.1.5.13 Unimplemented Registers

The following registers are defined in the PCI Specification or PCI System Design Guide, but are not implemented in Schizo's PBMs for the indicated reasons.

Cache Line Size - The cache line size is fixed at 64-bytes by the Safari architecture.

BIST - Built-In-Self-Test is not implemented in Schizo in a way that would need to be controlled via PCI Configuration registers.

Base Address Registers - The bridge itself has neither memory nor I/O space. It's configuration space is accessible only from the host and is hard-mapped.

Interrupt Line, Interrupt Pin - Do not apply. External PCI interrupt lines are handled by the RISC asic, while internal PBM interrupts are signalled directly to the Mondo block, and are not signalled on PCI interrupt lines.

Min_Gnt, Max_Lat - There is no regular traffic pattern to programmed I/O. Values of zero indicate there are no stringent requirements (true).

Disconnect Counter - This seems to be intended mainly for cases where the other bus (host bus in this case) is potentially very slow. This shouldn't apply to Safari.

Bridge Memory/IO Base and Limit Address - These registers are defined for an entirely flat address space which the Safari and Schizo cannot abide by.

DOS Attribute Registers - DOS compatibility is not a feature of Schizo.

4.4.2 IOMMU Registers

4.4.2.1 Translation Storage Buffer Overview

The IOMMU fetches translation information from a Translation Storage Buffer (TSB) in memory. The TSB contains one-level mapping information for the virtual DMA pages. A single TSB entry is called Translation Table Entry (TTE), and takes 8 bytes.

Schizo supports several TSB table sizes, specified by the TSB_SIZE field of IOMMU Control Register. TSB table sizes supported are 1K, 2K, 4K, 8K, 16K, 32K, 64K and 128K entries (not bytes), which allows a DVMA address space of 8M to 1G using 8K pages, and 64K to 2G using 64K pages (a DVMA address space larger than 2G is not supported, so 128K and 64K TSB sizes are not supported with a 64K page size). Software must set up TSB before it allows translation to start.

4.4.2.1.1 Translation Table Entry

Translation Table Entries (TTE) contain translation information for virtual pages. The IOMMU hardware reads one TTE during a table walk and stores it in the TLB. A TTE entry has valid information only when bit DATA_V is set. Information stored in the TTE has the following format:

Table 4-45 TTE Data Format

Field	Bits	Description
DATA_V	63	Valid bit (1 = TTE entry has valid mapping)
Reserved	62	Reserved.
DATA_SIZE	61	Page size of the mapping (0 = 8K, 1 = 64K)
STREAM	60	Stream bit (1 = streamable page, 0 = consistent page)
LOCALBUS	59	Local Bus bit. Not used
CONTEXT	58:47	Context number for this page, used in flushing multiple related pages.
Reserved	46:43	Reserved.
DATA_PA	42:13	Contains bits <42:13> of physical address. Bits 15:13 are not used for 64K page.
DATA_SOFT	12:7	Reserved for software use.
Reserved	6:5	Reserved.
CACHEABLE	4	Cacheable (1 = cacheable page, 0 = non-cacheable page)
Reserved	3:2	Reserved.
DATA_W	1	Set if this page is writeable
Reserved	0	Reserved.

Note – The LOCALBUS bit is not meaningful in a PCI environment, and is not stored in the TLB. The MMU hardware drops this bit after a table walk.

4.4.2.1.2 Tablewalk

During an IOMMU tablewalk (TSB lookup) the physical address for the TTE entry that will be fetched is calculated from the DVMA address (VirtAddr) as follows:

```
Table Size = 2 ^ (TSB_SIZE + 10)
Page Bits = (TBW_SIZE == 1) ? 15 : 12
Entry# = (VirtAddr >> Page Bits) & (Table Size - 1)
TTE Address = TSB_BASE + 8 * Entry#
```

The TSB is always accessed using cacheable Safari transactions.

4.4.2.1.3 Register Summary

Table 4-46 Addresses of IOMMU Registers

Register	Address	Access Size
IOMMU Control Register	PCI_CSRBase+0x00.0200	8 bytes
TSB Base Address Reg	PCI_CSRBase+0x00.0208	8 bytes
IOMMU Flush Page Register	PCI_CSRBase+0x00.0210	8 bytes
IOMMU Flush Context Register	PCI_CSRBase+0x00.0218	
TLB Compare Setup Diag Reg	PCI_CSRBase+0x00.A400	8 bytes
TLB Compare Result Diag Reg	PCI_CSRBase+0x00.A408	8 bytes
IOMMU LRU Queue Diag Regs	PCI_CSRBase+0x00.A500 - PCI_CSRBase+0x00.A57F	8 bytes
TLB Tag Diag Regs	PCI_CSRBase+0x00.A580 - PCI_CSRBase+0x00.A5FF	8 bytes
TLB Data RAM Diag Regs	PCI_CSRBase+0x00.A600 - PCI_CSRBase+0x00.A67F	8 bytes

4.4.2.2 IOMMU Control Register

The Control Register provides means to enable and disable the IOMMU and diagnostic mode, and set the TSB size and page size.

Table 4-47 IOMMU Control Register

Field	Bits	Reset	Description	Type
RESERVED	63:24	0	Reserved, read as zeros	R
LRU_LCKEN	23	0	LRU Lock Enable Bit. When set, only the TLB entry specified by the Lock Pointer can be replaced.	R/W
LRU_LCKPTR	22:19	X	LRU Lock Pointer. Works in conjunction with the LRU Lock Enable bit to limit TLB replacement to a single entry.	R/W
TSB_SIZE	18:16	X	TSB table size measured in the number of 8 byte entries. 0=1K, 1=2K, 2=4K, 3=8K, 4=16K, 5=32K, 6=64K, 7=128K.	R/W

Table 4-47 IOMMU Control Register

Field	Bits	Reset	Description	Type
RESERVED	15:3	0	Reserved, read as zeros	R
TBW_SIZE[1]	2	X	Assumed page size during TSB lookup. 0 = 8K page 1 = 64K page	R/W
MMU_DE	1	0	Diagnostic mode enable, when set it enables the diagnostic mode. See description of TLB tag diagnostics.	R/W
MMU_EN	0	0	IOMMU enable bit, when set it enables translations.	R/W

1. If DVMA mappings are always 8K pages, or mixed 8K and 64K pages, set this bit to '0' so that the index is constructed for 8K lookup. If all DVMA mappings are to 64K pages, set this bit to '1' so that the index is based on 64K pages. When this bit is '0', a 64K mapping should be placed in all 8 TSB entries in which it is indexed.

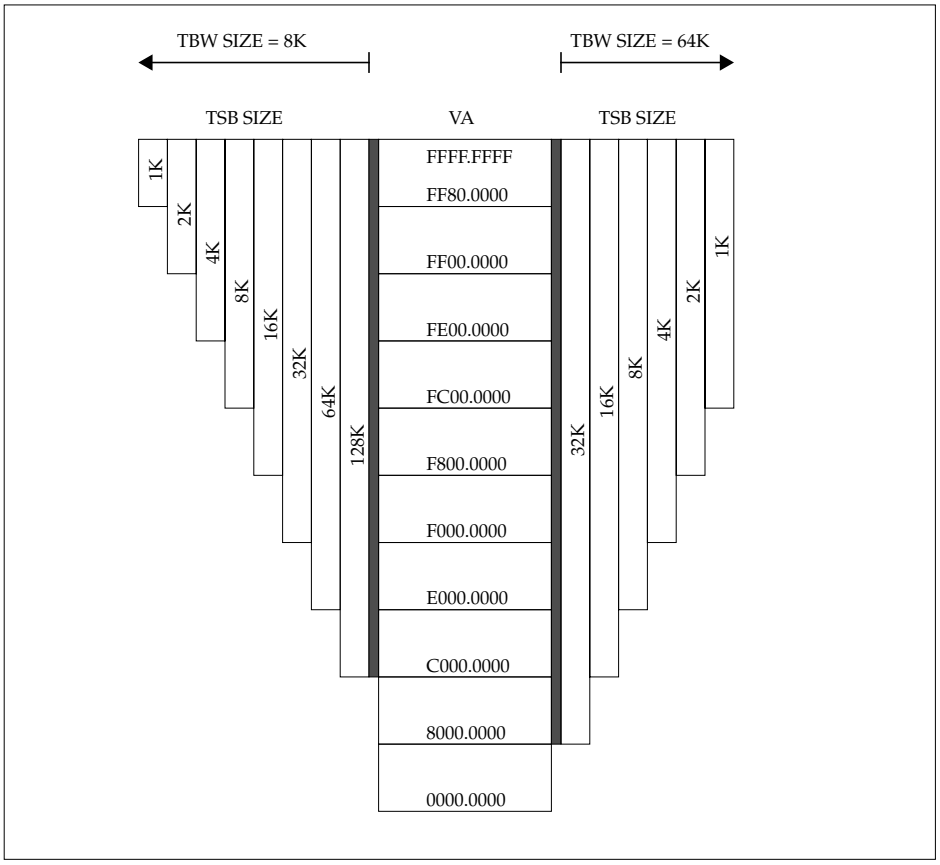
Table 4-48 Address space size and base address determination.

	TBW_SIZ == 0			TBW_SIZ == 1		
TSB_SIZE	VA Space Size	VA Base Address	TSB Index [3]	VA Space Size	VA Base Address	TSB_Index [3]
0	8 MB	0xFF80.0000	VA<22:13>	64 MB	0xFC00.0000	VA<25:16>
1	16 MB	0xFF00.0000	VA<23:13>	128 MB	0xF800.0000	VA<26:16>
2	32 MB	0xFE00.0000	VA<24:13>	256 MB	0xF0000000	VA<27:16>
3	64 MB	0xFC00.0000	VA<25:13>	512 MB	0xE000.0000	VA<28:16>
4	128 MB	0xF800.0000	VA<26:13>	1 GB	0xC000.0000	VA<29:16>
5	256 MB	0xF000.0000	VA<27:13>	2 GB	0x8000.0000	VA<30:16>
6	512 MB	0xE000.0000	VA<28:13>	not allowed[1]	--	--
7	1GB	0xC000.0000	VA<29:13>	not allowed[1]	--	--

1. Hardware does not prevent illegal combinations from being programmed. If an illegal combination is programmed into the IOMMU, all translation requests will be rejected as invalid.

Address space size and base address are controlled by TSB_SIZE and TBW_SIZ as shown in Table 4-48. Figure 4-1 shows the same information is a different format. Virtual addresses that are within Schizo's DVMA range (0x8000000-0xffffffff) on the PCI bus, but below the VA base address determined by the value of TSB_SIZE and TBW_SIZE are rejected by the IOMMU, and result in Target Aborts on the PCI bus

Figure 4-1 Legal DVMA address configurations.



4.4.2.2.1 *TLB locking*

For diagnostics and debugging, the IOMMU has the capability of restricting itself to use just a single entry of the TLB. This is controlled by the LRU_LCKEN and LRU_LCKPTR fields of the IOMMU Control Register. To properly turn locking on the following sequence is required:

- Set MMU_EN to 0
- Set LRU_LCKEN to 1 (must be a separate PIO write)
- Set LRU_LCKPTR to desired value (may be combined with previous PIO)
- Set MME_DE to 1 (may be combined with previous PIO)
- Invalidate all TLB entries
- Set MMU_EN to 1 and MMU_DE to 0.

To unlock the TLB:

- Set LRU_LCKEN to 0

4.4.2.3 TSB Base Address Register

The TSB Base Address Register contains the pointer to the first-entry of the TSB table. Together with part of the virtual address it uniquely identifies the address where hardware should fetch the TTE from the TSB table. The TSB table has to be aligned on 8K boundary. The lower order 13 bits are assumed to be 0x0 during TSB table lookup. Tables larger than 8K bytes are only constrained to be on 8K boundaries rather than having to be size aligned.

Table 4-49 TSB Base Address Register

Field	Bits	Reset	Description	Type
RESERVED	63:43	0	Reserved, read as zeros	R
TSB_BASE	42:13	X	Upper 30bits of the PCI TSB's physical address	R/W
RESERVED	12:0	0	Reserved, read as zeros	R

4.4.2.4 Flush Page Register

This is a write-only register that allows software to perform address based flushes of a mapping from TLB. The data written to this address contains the page number to be flushed. If there is a TLB entry with a matching page number, it will be invalidated.

Table 4-50 Flush Page Register

Field	Bits	Reset	Description	Type
Reserved	63:32	n/a	Reserved, write has no effect	W
FLUSH_VPN	31:13	n/a	31:16 = virtual page number if 64K page; bits 15:13 are don't care 31:13 = virtual page number if 8K page	W
Reserved	12:0	n/a	Reserved, write has no effect	W

Note – No hardware mechanisms exist to solve the potential race between a DVMA translation needing a TLB entry and the write to the Flush Page Register intended to flush that entry. Software must manage the interlock by guaranteeing that no DVMA can be going on to the page which is being flushed.

4.4.2.5 Flush Context Register

This is a write-only register that allows software to perform context based flushes of multiple mappings from TLB. The data written to this address contains the context number to be flushed. If there are any TLB entries with matching context numbers, they will be invalidated.

Table 4-51 Flush Context Register

Field	Bits	Reset	Description	Type
Reserved	63:12	n/a	Reserved, write has no effect	W
FLUSH_CONTEXT	11:0	n/a	Context number to flush. Context numbers have no inherent meaning to the IOMMU, but are used by software to group related pages.	W

Note – No hardware mechanisms exist to solve the potential race between a DVMA translation needing a TLB entry and the write to the Flush Context Register intended to flush that entry. Software must manage the interlock by guaranteeing that no DVMA can be going on to the context which is being flushed.

4.4.2.6 TLB TAG Diagnostics Access

The TLB Tag Diagnostics Access provides diagnostics path to the 16-entry TLB Tag when the MMU_DE bit in the IOMMU Control Register is turned on.

Table 4-52 TLB Tag Diagnostics Access

Field	Bits	Reset	Description	Type
RESERVED	63:37	0	Reserved, read as zeros	R
CONTEXT	36:25	X	Context Number	R/W
ERRSTS	24:23	X	Error Status: 00 = No Error 01 = Invalid Error 10 = Timeout 11 = ECC Error (UE)	R/W
ERR	22	X	When set to 1, indicates that there is an error associated with this TLB entry. The specific error is indicated by the ERRSTS field.	R/W
W	21	X	Writable bit. when set, the page mapped by the TLB has write permission granted.	R/W
S	20	X	Stream bit, 1 = page is streamable, 0 = page is not streamable	R/W
SIZE	19	X	Page Size, 0=8K and 1=64K.	R/W
VPN	18:0	X	VPN[31:13]	R/W

Note – Diagnostic accesses should insure that multiple page match conditions are not generated. The result of multiple page matches is unpredictable (multiple context matches are allowed)

4.4.2.7 TLB Data RAM Diagnostic Access

The TLB Data Diagnostics Access provides direct PIO accesses to 16 entries of TLB Data RAM. MMU_DE bit in the IOMMU Control Register must be turned on to perform the accesses. Following table shows the information included in the returned data.

Table 4-53 TLB Data RAM Diagnostics Access

Field	Bits	Reset	Description	Type
RESERVED	63:33	0	Reserved, read as 0.	R
V	32	0	Valid bit, when set, the TLB data field is meaningful.	R/W
RESERVED	31	0	Reserved, read as 0 (was local bus bit for SBus)	R/W
C	30	X	Cacheable bit. 1=Cacheable access, 0=Non-cacheable.	R/W
PA[42:13]	29:0	X	30-bit Physical Page Number.	R/W

4.4.2.8 LRU Queue Diagnostic Access

This LRU queue can be directly accessed by PIO read for diagnostic purpose. The MMU_DE bit in IOMMU Control Register must be set to perform direct access. There are 16 entries in the LRU Queue. Each entry contains a unique value range from 0x0 to 0x1F. Entry 0 contains the pointer to a TLB entry which is least recently used, and entry 15 contains the pointer to a TLB entry that is most recently used.

Table 4-54 LRU Entry Diagnostics Access

Field	Bits	Reset	Description	Type
RESERVED	63:4	0	Reserved, read as zeros	R
LRU_DO	3:0	X	LRU entry selected.	R

4.4.2.9 TLB Compare Setup Diagnostic Register

This register is used to set up the virtual address or context number for TLB compare diagnostics. The virtual address or context number is written to this register along with a flag indicating which one should be used, and the compare results from TLB can be read from the TLB Compare Result Diag Register.

Table 4-55 TLB Compare Setup Diagnostic Register

Field	Bits	Reset	Description	Type
RESERVED	63:33	0	Reserved, read as 0.	R
MATCH_TYPE	32	X	Set to 0 to select lookup by virtual page number, or 1 to select lookup by context number.	R/W

Table 4-55 TLB Compare Setup Diagnostic Register

Field	Bits	Reset	Description	Type
VPN	31:13	X	Virtual page number.	R/W
RESERVED	12	0	Reserved, read as 0.	R
CONTEXT	11:0	X	Context number for compare.	R/W

4.4.2.10 TLB Compare Result Diagnostic Access

Table 4-56 TLB Tag Comparator Diagnostics Access

Field	Bits	Reset	Description	Type
RESERVED	63:16	0	Reserved, read as zeros	R
COMP	15:0	X	TLB tag comparator output for each entry.	R

Note – The TLB Tag Comparator Diagnostics Access provides diagnostics path to the 16-entry TLB Tag Comparator when the MMU_DE bit in the IOMMU Control Register is turned on. Bit 0 represents the comparison result of the first TLB Tag entry, and bit 15 represents the last.

In order to avoid invalid address translation after TLB diagnostics, the valid bits in the TLB should be reset appropriately before doing any meaningful address translation.

4.4.3 Streaming Cache Operation

4.4.3.1 Streaming Cache Overview

The Streaming Cache (STC) implemented in Schizo is a small size fully associative cache managed by both hardware and software, to accelerate certain PCI bus DVMA to and from system memory.

Each PCI leaf within Schizo contains a STC module. This chapter specifies the size, functionality and algorithms of a single STC module, but the reader should realize that all statements can be applied to both copies of the STC.

The Schizo streaming cache performs three primary functions. The first is to accumulate sequentially addressed PCI write bursts into quantities the size of a system block. The second function is to speculatively prefetch the next (increasing) sequential block of memory for an active PCI read stream. The third function is to act as a local cache for PCI read accesses to the same block.

The implementation of the Schizo streaming cache features:

A fully associative pool of 16 entries shared among read and write streams.

Two 64-byte blocks of data per entry.

Dual ported data RAM for concurrent write/flush and read/fill operations.

64 bit wide interface to both the PBM and IFC modules

Least Recently Used entry allocation scheme

- Virtual address tags for low lookup latency.
- Physical address page translation for each entry to reduce flush and prefetch latencies.
- One entry allotment per virtual page to reduce the problem of individual misbehaved devices from thrashing the cache.
- Individual byte write enables to support PCI bus byte granularity.

Only accesses to virtual pages that are designated by software as streamable pages can use the streaming cache's functions. The streaming cache does not, however, participate in the system cache coherency protocol, and the data in the cache is out of the coherency domain, so software intervention is required to ensure a consistent memory image when transfers are complete. PCI devices will see program order functionality; reads followed by writes and writes followed by reads will see the correct results.

4.4.3.1.1 Streaming Cache Conceptual Overview

The streaming cache conceptually resides in close proximity to the PCI Bus Controller Module (PBM) so that low latency will be observed by the cards on each streaming access. A very tight coupling exists between the PBM and the streaming cache, with the STC almost appearing as a slave device to the PBM.

The streaming cache also has its own interface to the interface controller (IFC) of the PCI leaf. One reason is that the STC needs to pass different information to the IFC than the PBM. Another reason is that there are occasions when both the STC and the PBM would like to communicate with the IFC. The Schizo IFC can handle this arbitration more effectively than a PBM controlled arbiter since it has superior knowledge of the resources available.

IOMMU lookups are handled as a natural part of the PBM's functionality. When an entry is allocated in the STC, the PBM sends all necessary information, including the physical address for the page, therefore there is no direct connection between the STC and the IOMMU.

The STC is essentially an intermediate buffer for streaming data transfer between the Safari and PCI buses. There is no need for the STC to generate interrupts, therefore no connection to the Mondo Unit is necessary. Error information from Safari on read replies will be stored in the STC to be passed through to the PBM at a later time. PCI parity error information on write requests will be stored in the STC and transferred to the Safari interface, no corruption of ECC is performed for outbound data. It passes the data as is to the Safari core.

4.4.3.1.2 STC Subsections

The streaming cache is partitioned into four major subsections: the STC Central Control Unit, the Tag block, the Data block, and the Master Request Port.

The STC Central FSM handles contol signal communication between the STC and PBM module. It also, along with the Master request Port, handles STC communication with the PCI leaf's Interface Controller.

The Tag block contains all of the address and status information concerning the entries within the cache. This includes the virtual page number, the context number, the corresponding physical page number, the byte locations of dirty data within the data blocks, and the status of the line (invalid, fetching).

The Data block holds the actual data associated with the blocks of memory. It also contains error status in the form of 64 total bits, four bits per entry: one bit associated with PCI writes, and one bit associated with PCI reads for each of the two data blocks of that entry.

The Master Request Port handles most communication between the PCI leaf's internal bus structure and the streaming cache. It is responsible for posting DMA requests and subsequently sinking/sourcing data. To decouple streaming cache operation from the delays and protocol associated with using the PCI leaf's internal busses, the Master Request Port has both a prefetch buffer with four 64-byte entries, and a 64-byte flush buffer.

4.4.3.2 *Streaming Cache Functional Description*

The STC is first consulted whenever a PCI DMA request occurs. The virtual address appearing on the PCI bus is compared with the VA tags to see if the page is streamable and active in the cache. If there is a hit, then the appropriate action is taken on the matching entry as described in the following paragraphs. If the virtual page is not found in the STC, then this initial transaction will be ignored and the internal state reset. A page not found in the cache does not mean that the page is not streamable; just that it is not active in the cache at this time. The PCI controller module will return if the results of the IOMMU lookup indicate that the page was indeed marked streamable. In this case, the least recently used entry will be allocated and its contents invalidated (if clean) or flushed to memory (if the contents are dirty).

4.4.3.2.1 *Streaming Writes*

The STC receives write data from the PBM and fills the appropriate cache line in the hopes of eventually accumulating an entire system block quantity of data. This block can then be sent to memory without having to perform a read-merge-write operation. When a write crosses a block boundary, after the data is inserted into the cache, the completed (or partially completed) block will be copied into the 64-byte flush buffer, from which it will be sent to the Safari interface as soon as allowed by the IFC block. The streaming cache can continue the write in the other data block associated with the streaming cache entry (unless the block is the last one in the page). If the flush buffer is already full when a DMA write comes to a block boundary, the streaming cache will end the write on the block boundary, and will wait until the flush buffer begins emptying, then it will transfer the entry to the flush buffer. During this wait, no further requests from the PBM will be acknowledged. This guarantees that all completed 64-byte blocks will be flushed without software intervention.

4.4.3.2.2 *Byte Holes and Zero Byte Writes*

Byte holes within a single PCI write data stream (i.e. byte enable bit(s) is off, while byte enable(s) to the left and right are on), and zero byte writes are defined to be an error condition if the page is marked streamable. The PBM detects these conditions, sets a

status bit and signals an interrupt. The transaction continues, however, and the streaming cache beahves as if the byte holes didn't exist (i.e. it may overwrite data in memory that it shouldn't).

4.4.3.2.3 *Streaming Reads*

To improve streaming read performance, each streaming cache entry has storage for two separate 64-byte blocks of data.

Streaming reads fall into one of three categories; either the requested data is already located in the cache, the data must be fetched from memory, or the data is currently being fetched from memory.

If the data is in the cache, but the next consecutive block of data is not in the cache, and has not been prefetched already, a prefetch for that block is initiated if there is not already an outstanding prefetch. Then, if the last byte of the current block will not be read, the requested data is simply fed to the PCI device. If the last byte is expected to be read, as signalled by the PBM when it services a PCI Memory Read Line or Memory Read Line Multiple command, then a prefetch of the 2nd next sequential block of memory will also be attempted (the next block has already been prefetched, so this is the block after that), as long as there is not already an outstanding prefetch request. If the master request port is not busy, the prefetch will be launched immediately, otherwise the fetch will try to occur after the data is provided to the PCI device. When any prefetch data arrives, it is put into the prefetch buffer (which can hold four entries). As soon as the Data Block is not busy, any available prefetch data is copied to it, provided that the last word of the line being overwritten was really read by the PCI device. If the PCI device did not access the last word (even though it issued a Memory Read Line or Memory Read Line Multiple command), the prefetch (and some Safari bandwidth) will be wasted. A prefetch (again, of the 2nd next sequential block) can also be issued if the PCI device reads the last word of a line. This covers the normal PCI read case (i.e. the op is not a Memory Read Line or Memory Read Line Multiple).

If the data is not in the streaming cache (either the page or the line is not valid in the cache), then the data must be fetched from memory. The entry will be allocated by the fetch and marked as fetch outstanding. A demand fetch is then posted, after which a prefetch for the next sequential block is also issued. In this scenario, the PCI controller will be informed that the PCI device should be retried. This is done to free up the PCI bus to perform other transactions during the memory read latency.

If the PCI device hits on an entry in the cache that is currently marked fetch outstanding, the PBM will be informed that it should retry the PCI device, since the data is not immediately available.

When the end of a 64-byte block (that is not the end of an 8K page) is reached, the streaming cache can continue the read from the other data block in the entry, provided that the necessary data is already there. Otherwise it ends the transaction with the PBM.

4.4.3.2.4 *Entry Flushing*

There are several occasions when an entry containing dirty data needs to be flushed toward coherent memory. These are listed below.

- End of line flush: When a DVMA write reaches end of a cache line, the hardware will perform a line flush if the flush buffer is available, or wait until the flush buffer is available and then perform the flush.
- Non-sequential write to the same line: Any non-sequential write to a cache line will cause the existing line to be flushed before new data can be accepted.
- Line eviction on the same page: If a line is partially filled and is dirty and the device starts writing to a new line on the same page, buffered data for the previous write has to be flushed before new data can be accepted.
- Line eviction different page: This happens when all the cache lines are used up and the incoming DVMA accesses a page with no line allocated to it. If the LRU line has dirty data, it needs to be flushed before the line can be allocated to a new page.
- Read from page currently allocated for Writes: If a DVMA read occurs to a cache line currently being used for writes, and a line is dirty, then the buffer must be flushed before the read data is accepted.
- Software triggered flush: Software needs to flush the Streaming Buffer cache line on any DVMA write transfers which end on sub-line boundries. Software triggered flushes can be done either by virtual page number or by context number.

Of the above only the software triggered flush is visable to the software.

To make sure all previous flush operations are completed to the coherence domain, Schizo provides a mechanism to synchronize the flush operation. The flush synchronization involves a PIO write to the Streaming Buffer Flush Synchronization Register with the physical address of the flush flag provided as PIO write data. Only one write of the synchronization register is required as a barrier for all previous flush/invalidate writes to that streaming cache.

In all of the mentioned cases, it is possible that the entry may not contain an entire system block of data to be written to memory. In this situation, a partial block will be delivered to the Safari interface, which will then have to do a read/modify/write operation.

4.4.3.3 Streaming Cache Programming Model

4.4.3.3.1 Performance Issues

Extracting the most performance out of the streaming cache involves following several guidelines concerning DMA accesses. Not conforming to these guidelines will result in less than ideal observed performance. Some access patterns may, in fact, incur a penalty which causes the streaming cache to yield poorer performance than equivalent non-streamable accesses.

DMA writes to a block within a streamable virtual page should always access memory in increasing total sequential order (i.e. no gaps). Failure to do so will cause unnecessary flushing of the cache entry (and byte holes within a single DMA write will cause errors). Better performance is exhibited when writing large sized bursts (64 bytes or larger). While writes within a block should be increasing and sequential, no hardware imposed performance impact is made in regard to accesses across blocks or pages.

DMA reads from a block within a streamable virtual page should access memory in increasing sequential order, and should use the appropriate PCI bus commands based on amount of data to be read. Failure to do so will cause unnecessary prefetching . Reads across blocks should also be in increasing and sequential order. Failure to do so will waste the prefetching, reducing the maximum read bandwidth to no more than non-streaming reads. Prefetches are not launched if they would cross a page boundary. Larger burst sizes will again improve performance.

Since there is only one entry allotted per virtual page, multiple devices should not interleave their accesses to the same virtual page. If it is desired to have multiple devices accessing non-overlapping portions of the same page, aliasing should be used to map different virtual pages to the same physical page.

4.4.3.3.2 Streaming Cache Livelock Issues

Schizo is designed to perform special performance optimizations for DMA pages that are marked streaming. Performance on a streaming page works best when the actual DMA access pattern is “streaming” - that is, when the DMA transactions to that page access consecutively increasing addresses with no gaps, either within a transaction (via deassertion of PCI byte enables), or between transactions (by starting a new transaction somewhere other than where the previous one left off).

Normally, if the streaming guidelines are not followed, Schizo will still handle DMAs as expected, but with significantly poorer performance.

There is a scenario, however, where two (or more) DMA transactions can be presented to Schizo in a repeated fashion such that each transaction is always retried by Schizo, and never makes progress by having a data transfer.

The scenario requires two DMA reads that target the same streaming page, but which address different cachelines within that page. These two DMA read transactions could be from different devices on the same PCI bus, or they could conceivably be issued by the same device. Both transactions must be “active” on the PCI bus simultaneous - that is, they must have been issued, and must continue to be reissued when Schizo terminates them with a retry, and they must alternate on the PCI bus.

The first DMA read will cause Schizo to fetch data from memory to satisfy the DMA. Prior to obtaining the data, Schizo will retry either of the DMA read transactions as often as they are issued on the PCI bus. After Schizo has obtained the data, if the second DMA transaction shows up on the PCI bus prior to the first one, Schizo will throw away the data for the first DMA read and fetch data from memory to satisfy the second DMA (streaming DMA transactions to the same page share a common data buffer).

This process can now be repeated: when data for the second DMA arrives in Schizo, if the first DMA is the next transaction seen by Schizo, the data will be thrown away and a new data fetch started.

If the transaction continue to be alternated without significant variations in timing, this process can be repeated indefinitely. Neither DMA transaction makes any progress, and this situation can be called a livelock.

The only real workarounds for this problem are to avoid the situations that cause it. In particular, if a single device needs to exhibit this type of behavior, it should do its DMA in consistent mode (at least for affected pages). If that is not advisable for

performance reasons, the device will need to either keep retrying a single DMA read until data is transferred before moving to another section of the same streaming page, or it will need to manage the retries of the transactions in some other manner to avoid a pattern that causes indefinite livelock.

For the case where two separate devices are issuing the DMA reads, the same end result is required in order to work around the problem, but it is more difficult since the two devices probably can't coordinate activities with each other. In most cases this is not a supported configuration (DMA addresses given to one device driver should not be shared with a different driver), and the best solution is to map the page in consistent mode.

Note – This exists in Schizo 2.2 and also will not be fixed in future revisions of Schizo.

4.4.3.3.3 Memory Coherency Maintenance

The streaming cache resides outside of the coherent memory domain, therefore it must rely on software maintenance to guarantee correctness to the PCI devices. Two mechanisms have been defined to provide software access to the operation of the streaming cache.

The first means of software maintenance is either the "page invalidate/flush" or the "context invalidate/flush" command (henceforth referred to simply as flush commands). Software can issue a flush command to force the streaming cache to remove any one entry that matches the indicated virtual page or context number. If the page contains dirty data, a DMA write will be issued to flush the data to memory. Regardless of the contents of the entry, both the virtual page entry and its corresponding physical page translation will be invalidated from the cache. De-mapping a streamable page must involve flush command(s) to the streaming cache to ensure that virtual to physical page translations are deleted. When flushing by virtual page number, one flush is needed per page that is de-mapped. When flushing by context number, a separate flush is needed for each entry that matches the context, since each flush only removes one entry. A register is provided so that software can determine how many entries match a given context number.

Multiple flush commands are allowed to be outstanding at anytime. The PIO writes will be serviced in the order received by the cache.

Note – The streaming cache considers all pages to be 8K in size, therefore, when flushing by virtual page number, virtual pages to be flushed need to have bits 31 through 13 to be significant. When dealing with a 64K page size, eight separate virtual page flush commands will have to be issued to ensure that the entire 64K page is consistent between memory and the streaming cache.

The second mechanism provided for software to help maintain the cache is a means to synchronize with the DMA flush stream. The positive acknowledge to the PIO write command on a flush operation is not sufficient to indicate whether the data from the appropriate virtual page has been flushed all the way into the coherent memory domain. There is no ordering enforced between the PIO and DMA datapaths; the data from the cache entry can reside in an intermediate write buffer for an unknown period of time after the PIO acknowledge has been received by the processor. The "synch"

command has been created to properly inform software when the flush data has actually reached the coherent domain. The command is launched by issuing a PIO write to the streaming cache synch port with a (block aligned) physical address pointer. The cache will subsequently launch a DMA block write (data = 0x00000000 00000001, 0x0) to the supplied physical address to indicate that the flush operation has completed.

Since PIO accesses are serviced by the STC in the order received, and write requests delivered to the PCI leaf are strongly ordered, only one synch operation is needed to indicate that all previous flush commands for a particular streaming cache have completed and their data has entered the coherent memory domain.

4.4.3.3.4 Error Recovery

Whenever any DVMA read results in a Safari error (including uncorrectable ECC errors), the corresponding virtual page must be invalidated in the streaming cache. In some cases, software will only be provided with the physical address of the erroneous location. It is software's responsibility to determine the virtual page(s) that correspond to the error and subsequently invalidate them. Failure to conform to this procedure will result in a PCI device to be continuously error acked due to the presence of the error bit in the streaming cache entry. In the case of an uncorrectable ECC error, no further interrupts will be launched by the ECC unit, and the (stale) entry will most likely never be replaced in the cache (on it's own, by a prefetch) since it will be continually accessed and error ack'd.

Software will be notified via interuppt for DVMA write errors. These errors (which are the result of PCI parity errors) cause will not cause bad ECC to be written back to memory (normal data gets written).

4.4.4 Streaming Cache Registers

Table 4-57 Offset of Streaming Cache Registers

Register	Address	Access Size
Streaming Cache Control Reg.	PCI_CSRBase+0x00.2800	8 bytes
Streaming Cache Page Flush/Invalidate Reg	PCI_CSRBase+0x00.2808	8 bytes
Streaming Cache Flush Synchronization Reg	PCI_CSRBase+0x00.2810	8 bytes
Streaming Cache Context Flush/Invalidate Reg	PCI_CSRBase+0x00.2818	8 bytes
Streaming Cache Data RAM Diagnostic	PCI_CSRBase+0x00.B000 - PCI_CSRBase+0x00.B7FF	8 bytes
Streaming Cache Error Status Diagnostics	PCI_CSRBase+0x00.B800 - PCI_CSRBase+0x00.B8FF	8 bytes
Streaming Cache Page Tag Diagnostics	PCI_CSRBase+0x00.BA00 - PCI_CSRBase+0x00.BA7F	8 bytes
Streaming Cache Line Tag Diagnostics	PCI_CSRBase+0x00.BB00 - PCI_CSRBase+0x00.BB7F	8 bytes
Streaming Cache Context Match Reg	PCI_CSRBase+0x01.0000 - PCI_CSRBase+0x01.7FFF	8 bytes

4.4.4.1 Streaming Cache Control Register

This register controls the various functions of the selected streaming cache.

Table 4-58 Streaming Cache General Control Register

Field	Bits	Reset	Description	Type
Reserved	63:08	0	Reserved, read as 0	R
LRU_LPTR	7:4	0x0	LRU Lock Pointer. Works in conjunction with LRU_LE to restrict all streaming cache replacement operations to use a single entry.	R/W
LRU_LE	3	0	LRU Lock Enable. When set, only the entry specified by LRU_LPTR will be victimized.	R/W
RR_DIS	2	0	Rerun Disable. When set, the streaming cache will not rerun the PBM on check or put line misses.	R/W
DE	01	0	Diagnostic Mode enable, Set to 1 to enable diagnostic mode access.	R/W
SB_EN	0	0	Streaming cache enable/disable. Set to 1 to enable Streaming cache.	R/W

4.4.4.1.1 Streaming cache entry locking

For diagnostics and debugging, each STC has the capability of restricting itself to use just a single entry. This is controlled by the LRU_LE and LRU_LCKPTR fields of the STC Control Register. To properly turn locking on the following sequence is required:

- Set SB_EN to 0

- Set LRU_LE to 1 (must be a separate PIO write)
- Set LRU_LCKPTR to desired value (may be combined with previous PIO)
- Set DE to 1 (may be combined with previous PIO)
- Invalidate all STC entries
- Set SB_EN to 1 and DE to 0.

To unlock the STC:

- Set LRU_LE to 0

4.4.4.2 Streaming Cache Page Invalidate/Flush Register

This is a write-only register. It provides a means for software to cause an entry in the streaming cache with a matching virtual page to become invalidated/flushed (there should never be more than one entry matching a given virtual page). The data written to this address contains the virtual page number to be used for match comparison. The flush/invalidation is always based on 8K page size.

Table 4-59 Streaming Cache Page Invalidate/Flush Register

Field	Bits	Reset	Description	Type
FLUSH_A	31:13	n/a	8K virtual page to be invalidated/flushed	W
Reserved	12:0	n/a	These bits are ignored	W

4.4.4.3 Streaming Cache Flush Synchronization Register

The Flush Synchronization Register provides a means for software to determine when flush data has entered the coherent memory domain. Data written to this register contains the physical address of a flush flag. Writing to this register triggers Schizo to write a 64-byte block of data to FLAG_PA after all in progress flush operations (page based or context based) for the streaming cache are complete. The first doubleword of the block will be set to 0x1, and the remaining doublewords will be set to 0x0. The low order 6 bits of the FLAG_PA Address will be ignored. Please read the Streaming Cache chapter for more information of how the synchronization is done.

Table 4-60 Streaming Cache Flush Synchronization Register

Field	Bits	Reset	Description	Type
Reserved	63:43	n/a	Reserved. Write as 0.	W
FLAG_PA	42:6	n/a	64-byte aligned physical address for synch update.	W
Reserved	5:0	n/a	These bits are ignored.	W

4.4.4.4 Streaming Cache Context Invalidate/Flush Register

This is a write-only register. It provides a means for software to cause entries in the streaming cache with matching context numbers to become invalidated/flushed. Writing this register will cause all entries with a matching context number that do not have dirty data to be invalidated. In addition, one matching entry with dirty data will be flushed and invalidated. The data written to this address contains the context

number to be used for match comparison. Software can use this register along with the Streaming Cache Context Match Register to efficiently flush a context, and know when flushing is complete.

Table 4-61 Streaming Cache Context Invalidate/Flush Register

Field	Bits	Reset	Description	Type
Reserved	63:12	n/a	Reserved, ignored on write.	W
CONTEXT	11:0	n/a	Context number to flush/invalidate	W

4.4.4.5 *Streaming Cache Context Match Register*

The Streaming Cache Context Match Register is mapped at 4096 different locations. The register returns a bit vector for the 16 entries in the streaming cache, indicating which ones match a particular context number. It additionally returns a single bit indicating if there is at least one match in the cache. The context number for the comparison comes from bits <14:3> of the register address.

Table 4-62 Streaming Cache Context Match Register

Field	Bits	Reset	Description	Type
HIT	63	0	Set to 1 if at least one entry matches context	R
Reserved	62:16	0	Reserved, read as 0.	R
MATCH	15:0	0x0000	1 bit for each entry, 1 = context match.	R

4.4.4.6 *Streaming Cache Page Tag Diagnostic Access*

The Page Tags are directly accessible through PIO access. There is one register for each of the 16 entries in the streaming cache. This register can only be written when the DE bit of the Streaming Cache Control Register is set to 1.

Table 4-63 Streaming Cache Page Tag Format

Field	Bits	Reset	Description	Type
PTVD	63	0	Valid bit for page.	R/W
PTRD	62	1	Read (/write_) bit for page.	R/W
PTPA	61:32	X	Physical page number (as an 8K page)	R/W
PTVA	31:13	X	Virtual page number (as an 8K page)	R/W
Reserved	12	0	Reserved, read as 0.	R
CONTEXT	11:0	X	Context number	R/W

Caution – Valid bits on all entries should be reset to 0 after finishing diagnostics of the Page Tags.

Warning – Accessing diagnostic registers in Schizo's streaming cache block while DMA is active can occasionally cause one of two problems. Diagnostic PIO reads can cause Schizo to hang, leading to a system bus timeout, which will typically cause a

machine panic. Other diagnostic PIO accesses (reads or writes) can cause corruption of streaming DMA read data. Do not access Schizo's streaming cache diagnostic registers while a DMA is active.

4.4.4.7 *Streaming Cache Line Tag Diagnostic Access*

The Line Tags contains information related to the line in the streaming cache. There is one register for each of the 16 entries in the streaming cache. This register can only be written when the streaming cache is in diagnostic mode, (DE bit is set in the Streaming Cache Control Register).

Table 4-64 Streaming Cache Line Tag Format

Field	Bits	Reset	Description	Type
Reserved	63:27	0	Reserved, read as 0.	R
LRU	26:23	X	LRU index. Provides index of the streaming cache entry that would be allocated next at any given time. This is either the lowest numbered invalid entry, or the current LRU entry if all entries are valid.	R
LTVD1	22	0	Valid bit for block 1.	R/W
LTVD0	21	0	Valid bit for block 0.	R/W
LTFH1	20	0	Fetch Outstanding/Flush Necessary bit for block 1.	R/W
LTFH0	19	0	Fetch Outstanding/Flush Necessary bit for block 0.	R/W
LTSP	18:13	X	Start pointer for dirty data portion of current block. (only valid on page designated for writes where only one bank is specified.)	R/W
LTLAW	12:06	X	Line address for current block (Writes)	R/W
LTEPW	05:00	X	End pointer (+1) for dirty data portion of current block (Writes)	R/W
LTLA1	12:07	X	Line address[1] for bank 1	R/W
LTLA0	05:00	X	Line address[2] for bank 0	R/W

1. LTLA1 contains the most significant 6 bits of line address for bank1(e.g. address = {LTLA1[5:0],1'b1})
2. LTLA0 contains the most significant 6 bits of line address for bank0 (e.g. address = {LTLA1[5:0],1'b0})

The LTEPW is only valid when the page is writable. This field should be set to one greater than the end byte address of the dirty data chunk in the data ram (modulo block size of 64 bytes).

Similarly, LTLAW is also only valid when the page is writable. All 7 bits must be specified. If the page is readable, the 6 most significant bits of the line address must be specified for each banks (LTLA0/LTLA1). The least significant bit is determined by the bank number.

Caution – Valid bits on all entries should be reset to 0 after finishing diagnostics of the Line Tag.

Warning – Accessing diagnostic registers in Schizo's streaming cache block while DMA is active can occasionally cause one of two problems. Diagnostic PIO reads can cause Schizo to hang, leading to a system bus timeout, which will typically cause a machine panic. Other diagnostic PIO accesses (reads or writes) can cause corruption of streaming DMA read data. Do not access Schizo's streaming cache diagnostic registers while a DMA is active.

4.4.4.8 *Streaming Cache Data RAM Diagnostic Access*

There are sixteen entries in the streaming cache, each of which has two 64-byte blocks of data. All of this data is accessible via the Streaming Cache Data RAM Diagnostic registers. When accessing these registers, address bit <10:7> select the entry number, bit <6> selects which block of the entry and bits <5:3> select which 8-byte quantity to access in the block.

Table 4-65 Streaming Cache Data RAM Content Format

Field	Bits	Reset	Description	Type
DRDA	63:00	X	Data	R/W

Warning – Accessing diagnostic registers in Schizo's streaming cache block while DMA is active can occasionally cause one of two problems. Diagnostic PIO reads can cause Schizo to hang, leading to a system bus timeout, which will typically cause a machine panic. Other diagnostic PIO accesses (reads or writes) can cause corruption of streaming DMA read data. Do not access Schizo's streaming cache diagnostic registers while a DMA is active.

4.4.4.9 *Streaming Cache Error Status Diagnostic Access*

Each 64-byte block of data within the Streaming Cache has an error bit associated with it for DMA read operations and another error bit associated with DMA write operations. These bits are only visible to software during diagnostic mode. When accessing the Streaming Cache Error Status Diagnostic registers, address bits <7:4> select the entry number and bit <3> selects the block within the entry.

Table 4-66 Streaming Cache Error Status Format

Field	Bits	Reset	Description	Type
Reserved	63:2	0	Reserved, read as 0.	R
DWER	1	0	DMA write error (PCI parity error) bit.	R/W
DRER	0	0	DMA read error (UPA read reply error) bit.	R/W

Warning – Accessing diagnostic registers in Schizo's streaming cache block while DMA is active can occasionally cause one of two problems. Diagnostic PIO reads can cause Schizo to hang, leading to a system bus timeout, which will typically cause a machine panic. Other diagnostic PIO accesses (reads or writes) can cause corruption of streaming DMA read data. Do not access Schizo's streaming cache diagnostic registers while a DMA is active.

4.4.5 *Interrupt Registers*

PCI related interrupts are delivered to the processor by Schizo by way of the Safari Interrupt transaction type, and have the format shown in Table 4-67.

Table 4-67 Safari interrupt packet format

Interrupt packet Address/Data	Bits	Definition
Address	42:39	Reserved, issued as 0.
	38:34	Schizo NodeID
	33:29	Schizo AgentID
	28:24	Reserved, issued as 0.
	23:19	Target NodeID
	18:14	Target AgentID
	13:4	Reserved, issued as 0.
Data - DW0	63:16	Reserved, issued as 0.
	15:11	Source NodeID
	10:6	Interrupt Group Number (IGN)
	5:0	Interrupt Number Offset (INO)
Data - DW1-DW7	63:0	Reserved, issued as 0.

The IGN and INO field in the first doubleword of the interrupt data packet together make up an 11-bit interrupt number (INR), which indicates the source of the interrupt. Where possible, the interrupt is precise (i.e., it points to only one interrupt source). This permits the dispatch of the proper interrupt service routine without any register polling. Above the first doubleword the bits of the packet are guaranteed to be zero. Software can use this knowledge to distinguish these interrupts from others such as cross-calls.

In addition to PCI related interrupts, the PCI leaf within Schizo is responsible for generating some interrupts on behalf of other portions of the chip, including some internal sources (uncorrectable and correctable ECC errors) as well as external sources (UPA64S port interrupts).

Each PCI leaf contains identical logic, and is capable of generating the same set of interrupts. Before any particular interrupt becomes active, software must set a valid bit in the mapping register for that interrupt in one of the PCI leaf blocks. In general, there

is only one “correct” PCI leaf which should have the valid bit set for a given interrupt, although which leaf is “correct” can vary from system to system. If an interrupt is associated with a PCI device, then its valid bit should only be set in the PCI leaf that controls the bus where the device is located. This will guarantee that Schizo correctly synchronizes DMA write data with interrupt packets. If the wrong valid bit is set, DMA write data may still be buffered in Schizo when an interrupt is delivered. If both valid bits are set, two separate interrupt packets will be issued, one of which may pass related DMA write data. For interrupts that are not directly associated with PCI devices, software may choose which PCI leaf to use to deliver the interrupt packets.

There are 64 available INO values available to Schizo, which are broken down into groups according to Table 4-67.

Table 4-68 Interrupt Number Offset Assignments

INO (binary)	INO (hex)	Interrupt Source
0sssnn	00-1f	PCI Slot Interrupts, 4 separate interrupts per slot. sss = PCI slot number, 0-7 nn = 00 for INTA#, 01 for INTB#, nn = 10 for INTC#, 11 for INTD#
10nnnn	20-2f	On-board I/O (OBIO) interrupts
10101n	2a, 2b	UPA slot interrupts (included in OBIO category) Pulse type interrupts
110nnn	30-37	Internal interrupts, where nnn is defined as: 000 = UE (Uncorrectable ECC error) 001 = CE (Correctable ECC error) 010 = AE (PCI Bus A Error) 011 = BE (PCI Bus B Error) 100 = SE (Safari Bus Error) 101 = Reserved (BCDMA sync - see note) 110 = Reserved (ACDMA sync - see note) 111 = Reserved (unusable)
111nnn	38-3f	Reserved (unusable). Associated registers in Schizo do not exist.

Each PCI leaf has a set of registers for all of the possible interrupt sources. These registers are listed in Table 4-69

Table 4-69 Offset of Interrupt Registers

Register	Offset	Access Size
Interrupt Mapping Register for interrupt INO	PCI_CSRBase+ 0x00.1000+INO*8	8 bytes
UPA Port 0 Interrupt Mapping Register (register is also mapped normally via INO)	PCI_CSRBase+ 0x00.6000	8 bytes
UPA Port 1 Interrupt Mapping Register (register is also mapped normally via INO)	PCI_CSRBase+ 0x00.8000	8 bytes
Clear Interrupt Register for interrupt INO	PCI_CSRBase+ 0x00.1400+INO*8	8 bytes
PCI Int State Diag Register	PCI_CSRBase+0x00.A800	8 bytes

Table 4-69 Offset of Interrupt Registers

Register	Offset	Access Size
OBIO and Internal Int State Diag Register	PCI_CSRBase+0x00.A808	8 bytes
Interrupt Retry Register	PCI_CSRBase+0x00.1A00	8 bytes
PCI Consistent DMA Flush/Sync Register	PCI_CSRBase+0x00.1A08	8 bytes

4.4.5.1 *Interrupt Mapping Registers*

The format for each interrupt mapping registers is shown in Table 4-70

Table 4-70 Format of Interrupt Mapping Registers

Field	Bits	Reset	Description	Type
Reserved	63:32	0	Reserved, read as 0	R
V	31	0	Valid bit When set to 0, interrupt will not be dispatched to CPU from this PCI leaf. Has no other impact on interrupt state.	R/W
T_AgentID	30:26	X	Target AgentID of the processor that this interrupt will be sent to.	R/W
T_NodeID	25:21	X	Target NodeID of the processor that this interrupt will be sent to.	R/W
Reserved	20:11	0	Reserved, read as 0	R
IGN	10:6	-	Interrupt Group Number. This field always reflects the value of Schizo's AgentID (AId field in the Schizo Control/Status Register).	R
INO	5:0	-	Interrupt Number Offset The value of this field is hardwired for each mapping register, as shown in Table 4-69	R

4.4.5.2 *Interrupt Clear Registers*

Each interrupt source also has a state register associated with it. This state register can be either of type "level" or of type "pulse."

In the level sensitive case, the state register has two bits, and there are three valid states: IDLE, RECEIVED, and PENDING. IDLE represents the state where no interrupts are reported. RECEIVED indicates that an interrupt has been detected and should be delivered to the processor if/when the valid bit is set in its the mapping register. PENDING is the state when the interrupt has been queued to be or has been sent to the processor. Any subsequent detection of the same interrupt is ignored until software resets the state machine back to IDLE.

The state register for each level sensitive interrupts can be set to any desired state by software via the Clear Interrupt Registers.

Note – Software should ensure that the source of a level sensitive interrupt (i.e. the PCI device, or Schizo internal register) is cleared before clearing the interrupt state register via the Interrupt Clear Register, otherwise Schizo will continue to reissue the interrupt each time the state register is set to IDLE.

In the pulse case, the state register allows only two states: IDLE and RECEIVED. These states have the same meaning as for the level sensitive case. There is no PENDING state, so the state machine transitions from RECEIVED back to IDLE when the interrupt is dispatched to a processor. The only pulse type interrupts are the two UPA port interrupts.

The format for the Interrupt Clear Registers is shown in Table 4-71 and Table 4-72.

Table 4-71 Clear Interrupt Register - Level Interrupts

Field	Bits	Reset	Description	Type
reserved	63:2	n/a	Reserved. Write with 0.	W
State	1:0	n/a	State bits for the interrupt state machine associated with this interrupt. The following values may be written: 00 - Set state machine to IDLE state 01 - Set state machine to RECEIVED state 10 - Reserved 11 - Set state machine to PENDING state	W

Table 4-72 Clear Interrupt Register - Pulse Interrupts

Field	Bits	Reset	Description	Type
reserved	63:2	n/a	Reserved. Write with 0.	W
State	1:0	n/a	State bit for the interrupt state machine associated with this interrupt. The following values may be written: 00 - Set state machine to IDLE state 01 - Set state machine to RECEIVED state 10 - Reserved 11 - Reserved	W

Note – The Interrupt Clear Registers are write only. To determine the current interrupt state, use the interrupt state diagnostic registers instead.

4.4.5.3 *Interrupt State Diagnostic Registers*

The state bits for every interrupt source are made available in one of the two Interrupt State Diagnostic Registers. The state bits here have the same definitions as in Table 4-71.

The locations of each set of state bits is derived from the associated INO:

Register: if (INO >= 0x20) then OBIO Int Diag Reg else PCI Int Diag Reg

Bits: Int Diag Reg [(2 * (INO & 0x1F)) + 1: (2 * (INO & 0x1F))]

Table 4-73 PCI Interrupt State Diagnostic Register

Field	Bits	Reset	Description	Type
PCI_INT_STATE	63:0	0xFFFFFFFF.FFFFFFFF	State bits for PCI interrupts (INO 0x00-0x1f)	R

Table 4-74 OBIO and Internal Interrupt State Diagnostic Register

Field	Bits	Reset	Description	Type
Reserved	63:48	0x0000	Reserved, read as 0.	R
INT_INT_STATE	47:32	0xFFFF	State bits for Internal interrupts (INO 0x30-0x37).	R
OBIO_INT_STATE	31:0	0xFF0FFFFF	State bits for on-board I/O interrupts (INO 0x20-0x2f).	R

4.4.5.4 *Interrupt Retry Timer Register*

If an interrupt packet sent by Schizo is NACK'd by the Safari target agent, Schizo will wait for a programmed number of clocks and resend the interrupt. This register controls the number of clocks the interrupt dispatch unit should wait before resending the interrupt packet. The count specified by this register is not precise: there is a free running down counter (operating at the PCI leaf main frequency, nominally 66.7 MHz) which is loaded from this value and which must roll through 0 twice before the packet is retried.

Table 4-75 Interrupt Retry Timer Register

Field	Bits	Reset	Description	Type
RESERVED	63:20	0	Reserved, read as 0	R
LIMIT	19:0	0xFFFFF	Limit - the retry interval	R/W

Note – The Retry timer provides maximum of 1M clocks of delay before re-issuing the interrupt to Safari. The maximum delay is approximately 31.5 msec using the internal Schizo clock for a reference source (15.7 msec per iteration through the counter with a worst case of nearly two complete cycles counting to 0). The minimum delay is (count + 1) clock cycles.

4.4.5.5 *PCI Consistent DMA Flush/Sync Register*

The PCI 2.1 specification requires Schizo to flush any internal consistent DMA write buffers before issuing any PIO read on the PCI bus. Unfortunately, there are cases where this can cause a deadlock. For that reason, and for performance considerations, Schizo does not do this type of flushing.

This is normally not an issue. but, when there is a PCI-PCI bridge below Schizo that is not Sun4u compliant (most off the shelf PCI-PCI bridges will not be Sun4u compliant), there is a potential for DMA write data to not be flushed from Schizo when needed.

The PCI Consistent DMA Flush/Sync register allows software to do the necessary flushing manually in this case. When this register is written with a physical address, Schizo will first flush any consistent mode DMA write buffers that have data. Then Schizo will issue another DMA write to the specified physical address. It will write 64 bytes of data (the sync flag) to this address. The first doubleword of the data will be 0x0000000000000001. The remaining doublewords will be all 0. When the sync flag is seen in memory, all appropriate DMA write data is guaranteed to be in memory as well.

Table 4-76 PCI Consistent DMA Flush/Sync Register

Field	Bits	Reset	Description	Type
Reserved	63:43	n/a	Reserved, write as 0	W
FLAG_PA	42:6	n/a	64-byte aligned physical address to which the sync flag value will be written	W
Reserved	5:0	n/a	Reserved, write as 0	W

Note – The PCI 2.1 Spec requires a PIO read of a device in order to guarantee that data in intermediate PCI-PCI bridges is flushed. Schizo's PCI Consistent DMA Flush/Sync must be performed after this device PIO read, or DMA write data will not be properly flushed.

4.4.6 PCI Performance Monitor Registers

Table 4-77 Offset of PCI Performance Monitor Registers

Register	Offset	Access Size
PCI Performance Monitor Control Register	PCI_CSRBase+0x00.0100	8 bytes
PCI Performance Counter Register	PCI_CSRBase+0x00.0108	8 bytes
PCI Configuration/Idle Check Diagnostics Register	PCI_CSRBase+0x00.0110	8 bytes

In order to gather useful statistics on the PCI performance of Schizo, a pair of registers provide counts of key events. There are only two counters present, and the control register selects the input for each of the counters.

In addition, a diagnostic register that indicates whether various subsystems within the PCI leaf are idle is available, (this register also has some PCI configuration bits in it, see bit descriptions in Section 4.4.6.3, on page 97 for details).

4.4.6.1 *Performance Monitor Control Register*

This register controls the events to be monitored by the Performance Counter Register.

Table 4-78 PCI Performance Monitor Control Register

Field	Bits	Reset	Description	Type
Reserved	63:16	0	Reserved, read as 0	R
SEL1	15:11	0	Select event source for counter 1. See Table 4-79 for encoding.	R/W
Reserved	10:9	0	Reserved, read as 0.	R
SEL0	8:4	0	Select event source for counter 0. See Table 4-79 for encoding.	R/W
Reserved	3:0	0	Reserved, read as 0	R

The table below defines the encoding for selecting PCI events to monitor in Schizo.

Table 4-79 PCI Performance Counter Event Sources

SEL0, SEL1	Event Sources
0x00	Number of streaming DVMA read transfers
0x01	Number of streaming DVMA write transfers
0x02	Number of consistent DVMA read transfers
0x03	Number of consistent DVMA write transfers
0x04	Number of streaming buffer misses
0x05	Number of bus cycles PCI bus is busy with DMA.
0x06	Number of words transferred using DMA on PCI bus[1]
0x07	Number of bus cycles PCI is busy with PIO·
0x08-0x0f	Reserved
0x10	Number of TLB misses
0x11	Number of interrupts issued
0x12	Number of interrupt NACKs received
0x13	Number of PCI bus PIO read transfers
0x14	Number of PCI bus PIO write transfers
0x15	Number of consistent DMA read buffer timeouts
0x16	Number of PCI DMA read requests retried due to STC
0x17	Number of PCI DMA write requests retried due to STC
0x18	Number of PCI DMA read requests retried, not due to STC
0x19	Number of PCI DMA write requests retried, not due to STC
0x1a	Amount of time spent in E* slow mode (1 count for every 64 slow clock cycles)
0x1b	Number of times E* slow mode is entered (dynamically)
0x1a-0x1f	Reserved. Counter value is undefined when these sources are chosen.

1. The word count will increment whenever any of the four associated byte enables is active. If, during the recording interval, there are any DMAs that start or end on unaligned addresses, it won't be possible to determine the exact number of bytes transferred.

4.4.6.2 PCI Performance Counter Register

This register contains the counts of the two events selected by the PCI Performance Monitor Control Register. The two counters operate independently, but must both be written together. When either counter reaches its maximum count, it will silently wrap around to 0x0 and continues counting. If necessary, software must detect and handle this overflow condition.

Table 4-80 PCI Performance Counter Register

Field	Bits	Reset	Description	Type
CNT1<31:0>	63:32	0	Contains value for event counter 1.	R/W
CNT0<31:0>	31:00	0	Contains value for event counter 0.	R/W

4.4.6.3 PCI Configuration/Idle Check Diag Register

This register has two functions. It has some mode and enable bits. It also has an idle check diagnostic function. Each bit gives an indication of whether the indicated block is currently busy, or whether all of its resources and state machines are currently idle (outside of those that may be needed to support a PIO access to the Idle Check Diag Register). The exact definition of idle for each block will not be defined here, and is up to the hardware specification for each individual block.

Table 4-81 PCI Configuration/Idle Check Diag Register

Field	Bits	Reset	Description	Type
Reserved	63:16	0	Reserved, read as 0	R
PCI 2.0 Compatibility	15	0	When set, removes the PCI command from delayed read completion. This allows devices which change commands when retried, i.e. not PCI 2.1 compliant, to not timeout.	R/W
DMAW parity err interrupt enable	14	0	When set, DMA write parity errors will generate an interrupt. * bit 14 = 0 => 2.3 style behavior, don't interrupt on DMA writes with parity errors. * bit 14 = 1 => 2.4 style behavior, interrupt on any DMA writes with parity errors. * Power on reset value is 0., defaults to 2.3 style behavior, don't interrupt on DMA writes with parity errors.	R/W
Reserved	13:8	0	Reserved	R/W

Table 4-81 PCI Configuration/Idle Check Diag Register

Field	Bits	Reset	Description	Type
Reserved	7:5	0	Reserved, read as 0	R
IFC_NOT_IDLE	4	0	IFC block is not idle	R
MDU_NOT_IDLE	3	0	MDU block is not idle	R
MMU_NOT_IDLE	2	0	MMU block is not idle	R
PBM_NOT_IDLE	1	0	PBM block is not idle	R
STC_NOT_IDLE	0	0	STC block is not idle	R

For each block, a value of 0 indicates the block is idle, and a value of 1 indicates that the block considers itself busy.

Error Handling 5

5.1 Overview

This chapter describes the error detection, correction, and error reporting mechanisms supported by Schizo.

Detected errors are classified as either fatal errors or non-fatal errors. Fatal errors may result in a system reset if enabled by software. Actions taken on non-fatal errors include: generating interrupts, setting status register bits, or none at all.

5.2 Error Detection and Reporting

5.2.1 Error Detection

5.2.1.1 Detectable Safari Bus Errors

5.2.1.1.1 Safari Address Parity Error

This error is detected when the Safari AddrPty signal does not match the calculated parity of the Transaction Request Group signals from the previous cycle. This error can be detected on any Safari transaction that is not initiated by Schizo, regardless of whether Schizo was the intended target. Schizo will execute the command and checks and reports parity independently. The transaction is not explicitly ignored and may propagate through Schizo with a corrupted command/address/mask or atransid depending on which signal has the parity error. However if the S_ERROR_L reporting signal is enabled, the system will result in a reset resulting from the detection of the parity error.

5.2.1.1.2 Safari Unmapped Error

This error is detected when the MappedIn signal of the Safari State group is not asserted for a transaction that Schizo has initiated. This can happen for any of the following:

- Cacheable or non-cacheable DMA reads or writes
- Interrupt requests generated by Schizo

5.2.1.1.3 *Safari Timeout Error*

This error is detected if the DStat for an incoming data packet indicates a Timeout Error. This can happen for the following transaction types:

- Non-cacheable DMA reads

5.2.1.1.4 *Safari Bus Error*

This error is detected if the DStat for an incoming data packet indicates a Bus Error. This can happen for the following transaction types:

- Non-cacheable DMA reads

5.2.1.1.5 *Safari DStat Error*

This error is detected if the DStat for an incoming data packet is illegal. The following combinations are illegal:

- PIO write to Schizo or devices controlled by Schizo (except for UPA64S devices), with a Timeout Error or Bus Error indicated in the DStat field
- DMA read of cacheable memory, with DStat other than ECC valid
- Partial-line DMA write of cacheable memory, with DStat other than ECC valid

5.2.1.1.6 *Safari Datapath ECC Error*

This error is detected if the DStat for an incoming data packet indicates that ECC has been generated, and the ECC indicates an error. There are two varieties of this error that can be detected: correctable ECC errors, and uncorrectable ECC errors. These errors can happen when data is received by Schizo for any of the following types of transactions:

- PIO writes to Schizo and devices controlled by Schizo. (Note error logging is partial for UPA64S devices.)
- Cacheable or non-cacheable DMA reads
- Partial-line DMA writes to cacheable memory

When Schizo “owns” data it received with bad ECC, it will “force” bad ECC on data it returns. This occurs for partial-line DMA writes, which require writebacks and maybe copybacks. The data driven by Schizo in this case will be all zeroes, with the two least significant ECC check bits inverted.

5.2.1.1.7 *Safari Command Timeout Error*

Safari protocol requires outstanding commands be protected (against HW failures) by timeout counters. When a command at the head of a queue doesn’t complete within the timeout interval, a status bit will be set and S_ERROR_L will be asserted. The system may then provide the missing data or assert reset to unhang Schizo.

5.2.1.1.8 *Safari SSM Error*

Received an SSM command while SSM_EN is not set.

5.2.1.2 Detectable PCI Bus Error

5.2.1.2.1 PCI Receive Data Parity Error

This error is detected when a parity error is detected on either of the two sets of parity protected signals on the PCI bus (standard 32-bit data/control, and 64-bit data/control extensions), only during a valid data phase of a transaction (IRDY# and TRDY# asserted) involving Schizo. Note that parity errors will not be detected during wait states (neither master nor target wait states). PCI receive data parity errors can be detected during the following transactions:

- PIO reads from a PCI device
- DMA writes by a PCI device

5.2.1.2.2 PCI Send Data Parity Error

This error is detected when the PCI PERR# signal is asserted by the other device during a PCI transaction involving Schizo. This can be detected during the following transactions:

- PIO writes to a PCI device
- DMA reads by a PCI device

5.2.1.2.3 PCI Target-Abort

This error is detected when, during a PCI transaction for which Schizo is the master, the target device asserts STOP# and deasserts DEVSEL#. Reasons that a target might issue a target-abort include: unsupported byte enables, an unsupported addressing mode, an address parity error, and device specific errors. A target-abort may be detected during any PIO reads or writes to PCI devices.

5.2.1.2.4 PCI Master-Abort

This error is detected when Schizo begins a PCI transaction, and no device responds by asserting DEVSEL# within X cycles. This is typically because no device is mapped at the specified address. A Master-Abort error can be detected during any PIO read or write attempt to a PCI device.

5.2.1.2.5 PCI Retry Limit Error

This error is detected when Schizo initiates a PCI transaction, and the PCI target device terminates the transaction with a Retry (disconnects with no data transfer) for XXXX successive attempts. This error can be detected during any PIO read or write to a PCI device.

5.2.1.2.6 PCI Retry Timeout

When Schizo is the target of a PCI transaction that is determined to be a Consistent-mode DMA read, and Schizo turns it into a Delayed Read Transaction (terminates on the PCI with Retry, but reserves resources, and issues the request to Safari), and the PCI master does not re-issue the same transaction within 2^15 clocks after the data has been returned to the PCI leaf.

5.2.1.2.7 PCI Address Parity Error

This error is detected when a parity error is detected on either of the two sets of parity protected signals on the PCI bus (standard 32-bit data/control, and 64-bit data/control extensions). Schizo only detects address parity errors if the address seen by Schizo is a DMA address (i.e. bit 31 is set for normal addressing, or bits 63:50 are set for a dual-address cycle).

5.2.1.2.8 PCI System Error (SERR#)

This error is detected any time that the PCI SERR# signal is asserted by any PCI device. Possible reasons for a device to assert SERR# include: detection of an address parity errors, or device specific fatal errors.

5.2.1.3 Detectable UPA64S Errors

5.2.1.3.1 UPA64S Slot Vacant

When a UPA64S port which was detected to be unoccupied at the end of system reset, the address comparator for that slot is disabled. A PIO read or write directed to the empty slot will be ignored, causing the source to detect an unmapped error.

5.2.1.4 Other Detectable Errors

5.2.1.4.1 IOMMU Translation Error

This error can be detected during the address translation of any DMA read or write transaction. It is detected if any of the following occur:

- The PCI virtual address is out of the programmed DMA address range
- The Valid bit is not set in the TTE for the given virtual page
- The DMA operation is a write, and the TTE is marked read-only

5.2.2 Error Reporting

5.2.2.1 Summary of Error Reporting

Table 5-1 summarizes the response of Schizo when an error is detected. All Schizo operations involve the Safari bus and one other leaf bus or leaf block. Errors typically occur and are detected on a single bus, but error responses may occur on neither, one,

or both of the involved buses. Only the immediate response of Schizo is shown below, and not that of any other devices that may be involved. For example, when a Safari Data ECC Error is detected during a DMA Read from a PCI device, the table shows that Schizo issues a Target-abort, but does not indicate that the PCI device then typically issues an interrupt as a result. In all cases where Schizo responds with an interrupt, the interrupt is only issued if enabled.

The following register abbreviations are used in the table. See Chapter 4, “Register Descriptions” for further details:

- SIB CSR - Safari Interface Block Error Control/Log Register
- CE/UE AFR - Correctable/Uncorrectable ECC Error Address Fault Register
- PCI Status - PCI defined Status Register in PCI Config Space
- PCI CSR - PCI Block Control/Status Register
- PCI AFR - PCI Asynchronous Fault Register

Table 5-1 Summary of Error Reporting

Src/Dst Bus/Leaf	Transaction Type	Error Type	Safari Response	Leaf Bus Response	Error Status Register(s)
Unknown	Unknown	Safari Address Parity Error	Execute transaction, reports error via S_ERROR_L if enabled.	None	SIB CSR
PCI	DMA Read	Safari Unmapped Error	Transaction aborted	Target-abort	SIB CSR PCI Status
PCI	DMA Write	Safari Unmapped Error	Transaction aborted	None	SIB CSR
PCI leaf	Interrupt	Safari Unmapped Error	Transaction aborted	ACK interrupt	SIB CSR
PCI	DMA Read	Safari Timeout Error	None	Target-abort	SIB CSR PCI Status
PCI	DMA Read	Safari Bus Error	None	Target-abort	SIB CSR PCI Status
PCI	PIO Write	Safari DStat Error	None	Not passed to leaf	SIB CSR
PCI	DMA Read	Safari DStat Error	None	Target-abort	SIB CSR PCI Status
PCI	DMA Write (partial line)	Safari DStat Error	bad ECC forced on Writeback	None	SIB CSR
PCI	PIO Write	Safari Data ECC Correctable Error	CE Interrupt	None	CE AFR
PCI	DMA Read	Safari Data ECC Correctable Error	CE Interrupt	None	CE AFR
PCI	DMA Write (partial line)	Safari Data ECC Correctable Error	CE Interrupt	None	CE AFR

Table 5-1 Summary of Error Reporting

Src/Dst Bus/Leaf	Transaction Type	Error Type	Safari Response	Leaf Bus Response	Error Status Register(s)
PCI	PIO Write	Safari Data ECC Uncorrectable Error	UE Interrupt	Not passed to leaf	UE AFR
PCI	DMA Read	Safari Data ECC Uncorrectable Error	UE Interrupt	Target-abort[1]	UE AFR PCI Status
PCI	DMA Write (partial line)	Safari Data ECC Uncorrectable Error	bad ECC forced on Writeback; UE Interrupt	None	UE AFR
PCI	???	Safari SSM Error	None?	???	SIB CSR
PCI	None	PCI System Error	PCI Interrupt	None	PCI Status PCI CSR
PCI	Unknown (DMA)	PCI Address Parity Error	PCI Interrupt	Target-abort	PCI Status
PCI	PIO Read	PCI Master-abort	DStat=Timeout	None	PCI Status
PCI	PIO Write	PCI Master-abort	PCI Interrupt	None	PCI Status PCI AFR
PCI	PIO Write (PCI Special Cycle)	PCI Master-abort (This is the normal termination for PCI Special Cycles)	None	None	None
PCI	PIO Read	PCI Retry Limit Error	DStat=Timeout	Stop retrying	PCI Status
PCI	PIO Write	PCI Retry Limit Error	None	Stop retrying	PCI Status PCI AFR
PCI	PIO Read	PCI Target-abort	DStat=Bus Error	None	PCI Status
PCI	PIO Write	PCI Target-abort	PCI Interrupt	None	PCI Status PCI AFR
PCI	DMA Read	PCI Retry Timeout	PCI Interrupt	Free delayed transaction resources	PCI CSR PCI AFR
PCI	DMA Write, Streaming	PCI Receive Data Parity Error	Normal Data transfer to memory, PCI Interrupt	Assert PERR#	PCI Status
PCI	DMA Write, Consistent, STOP_DATA=0	PCI Receive Data Parity Error	Normal Data transfer to memory, PCI Interrupt	Assert PERR#	PCI Status
PCI	DMA Write, Consistent, STOP_DATA=1	PCI Receive Data Parity Error	Does not pass data to memory, PCI Interrupt	Assert PERR#	PCI Status
PCI	PIO Write	PCI Send Data Parity Error	PCI Interrupt	None	PCI Status PCI AFR

Table 5-1 Summary of Error Reporting

Src/Dst Bus/Leaf	Transaction Type	Error Type	Safari Response	Leaf Bus Response	Error Status Register(s)
PCI	PIO Read	PCI Receive Data Parity Error	DStat=Bus Error	Assert PERR#	PCI CSR PCI Status
PCI	DMA Read	PCI Send Data Parity Error	None	None	PCI Status
PCI	DMA Read	IOMMU Translation Error	PCI Interrupt	Target-abort	PCI Status MMU CSR
PCI	DMA Write	IOMMU Translation Error	PCI Interrupt	Target-abort	PCI Status MMU CSR
UPA64S	PIO Read	UPA64S Slot Vacant Error	(don't assert Mapped)	None	
UPA64S	PIO Write	UPA64S Slot Vacant Error	(don't assert Mapped)	None	

1. The Target-abort is signalled on the PCI bus when the bad data is actually requested there. In the case of a streaming cache prefetch, this may never occur. In addition, Schizo reserves the right to consider an entire data packet bad even when only a portion of it has an uncorrectable ECC error.

5.3 *Undetected Errors*

Certain error conditions are not detected or reported by Schizo. Examples of these errors are listed below. Please note that this list may not enumerate all unreported errors;

- A PIO write to a read-only register is ignored.
- A read from a write-only register returns unknown data.
- Certain PCI protocol violations (device responding with DEVSEL#, but never responding with TRDY# or STOP#).
- Infinite NACKing of an interrupt
- Safari DStat Errors on PIO writes to UPA64S